University of Zurich^UZH

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Passive Non-Intrusive User Profiling in ZigBee Smarthomes

Franziska Geiger
Zurich, Switzerland
Student ID: 18-706-283

Supervisor: Katharina O.E. Müller, Prof. Dr. Bruno Rodrigues
Date of Submission: May 14, 2025

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 14.05.2025

[illegible]

Signature of student

Abstract

Smart home devices are becoming increasingly pervasive, introducing growing privacy risks. Previous research has shown that encrypted ZigBee communication from systems like Philips Hue lights can be passively intercepted and inferred. However, the implications of such data for behavioral profiling and long-term habit tracking remain largely unexplored. Existing studies often focus on broader smart home ecosystems without implementing dedicated profiling systems for a single platform. This work presents a user profiling system focused exclusively on Philips Hue lighting devices, extending a passive sniffing application to infer user behavior. The system detects daily routines, identifies recurring habits, and analyzes long-term trends and anomalies through rule-based methods. Results show that passive ZigBee traffic alone can reveal user routines and behavioral deviations, even without decryption, demonstrating significant privacy risks.

Acknowledgments

I would like to thank the Communication Systems Group at the Department of Informatics of the University of Zurich for the opportunity to conduct my bachelor's thesis within their team. I am particularly grateful to my supervisor, Katharina Müller, for her valuable guidance, consistent support, and dedicated time throughout this project. I would also like to thank my parents for their relentless support and patience throughout this journey. Last but not least, I want to thank my best friend for always believing in me.

Contents

Chapter 1

Introduction

In recent years, the increasing demand for a more intelligent and connected lifestyle has driven the growth of consumer Internet of Things (IoT) devices [1]. The number of IoT connections in 2033 worldwide is forecasted to reach 39.6 billion, more than double the number we can count today (18 billion) [2]. Among IoT technologies, smart home devices have emerged as a prominent sector in the IoT landscape, offering significant advantages such has enhanced home security, increased energy efficiency and improved convenience for users [3].

While these benefits enhance everyday life, IoT devices also introduce significant security and data privacy concerns by continuously collecting a large amount of user activity data and private user information [1]. By analyzing data such as device usage patterns or communication signals, attackers can infer sensitive information about users' daily lives, including their routine, habits [4], and even their presence or absence at home [1]. Such insights could not only compromise user privacy but also pose a physical security threat, such as burglary. One standard communication protocol in IoT smart home devices is ZigBee, which will be the focus of this thesis.

ZigBee is a wireless communication protocol specifically developed to meet the demand for low-cost implementation of low-data-rate wireless networks with extremely low power consumption [5]. This makes it ideal for (battery-powered) smart home applications [5]. ZigBee has more relaxed requirements than other standards like IEEE 802.11, reducing complexity and making ZigBee-compliant transceivers cheaper to develop and implement [5]. However, the ease of implementing ZigBee comes with a trade-off, particularly regarding security and the vast amounts of data these devices collect. Because of its open design and low complexity, ZigBee allows attackers to passively intercept device traffic over the air, making it easy for attackers to capture ZigBee data and analyze it for sensitive user information [1].

1.1 Motivation

As the use of IoT devices continues to grow, this thesis aims to investigate the privacy vulnerabilities associated with ZigBee-enabled smart home systems, specifically focusing on how passive, non-intrusive techniques can be used for user profiling and long-term habit tracking.

As shown in a previous bachelor's thesis by [6], ZigBee network events could be inferred without decryption by developing a real-time scanning and sniffing application. The goal of this thesis is to explore how the inferred ZigBee data can be analyzed to create user profiles and track long-term habits. It will investigate the extent to which user profiling can compromise privacy, revealing critical vulnerabilities in ZigBee-enabled smart home systems.

The motivation for this work lies in the growing importance of addressing privacy concerns in IoT systems. As smart homes become more integrated into every day life, understanding the potential misuse of seemingly harmless device data becomes irremissible. By shedding light on the risks associated with passive eavesdropping and demonstrating the implications for user privacy, this thesis aims to raise awareness and inform future security improvements for ZigBee-based systems and similar IoT protocols. The following research questions will be addressed in this thesis:

1. How can ZigBee-network data be analyzed for passive, non-intrusive user profiling in smart home environments?
2. What are the limitations and possibilities of long-term habit tracking using ZigBee-network data, and how do they differ from short-term user profiling?
3. What privacy risks arise from the ability to infer user behavior and habits from ZigBee-network data?

1.2 Thesis Goals

This thesis aims to explore the potential of user profiling and long-term habit tracking using data derived from ZigBee communication events within the Philips Hue smart home ecosystem. Building on the real-time scanning and sniffing application developed by [6], this thesis seeks to extend its capabilities to include data analysis for inferring user behavior patterns and daily routines to Philips Hue devices. Additionally, this research will investigate current privacy frameworks and related work to contextualize the implications of these vulnerabilities within the broader landscape of IoT security and privacy. Therefore, the main contributions of this work are:

- An analysis of existing privacy frameworks and evaluation of how they address the challenges posed by user profiling and long-term habit tracking in Philips Hue smart home systems.
- To design and prototype tools for visualizing and interpreting the inferred user data for user profiling and behavioral trends within a Philips Hue environment.
- Evaluation of the privacy risks associated with user profiling and habit tracking by conducting experiments using Philips Hue devices, assessing the implications for privacy.

1.3 Thesis Outline

In Chapter 2, a background on ZigBee, Philips Hue, User Profiling, and Data Security and Privacy is provided. The chapter concludes with a review of related work, examining previous research in relation to ZigBee, Philips Hue, user profiling, and privacy risks.

Chapter 3 focuses on the design of the user profiling system by outlining different possible approaches and techniques and explaining the reason behind the chosen design.

Chapter 4 describes the implementation of the prototype, presenting the development decisions made for each script.

Chapter 5 evaluates the prototype using real-world sniffing data. It discusses the prototype's limitations, assesses privacy risks, and outlines challenges encountered during implementation.

Chapter 6 summarizes the key findings and reflects on the implemented user profiling system. It concludes by highlighting potential future research and system enhancements.

Chapter 2

Fundamentals

2.1 Background

2.1.1 ZigBee

In this section, basic information about the ZigBee protocol including the protocol structure, device types, and ZigBee Light Link (ZLL) are introduced.

ZigBee Structure

ZigBee is a wireless communication protocol targeted for battery-powered applications where low data rate, low cost, and long battery life are the most important requirements [5]. The ZigBee protocol architecture consists of the physical layer (PHY), the medium access control layer (MAC), the network layer (NWK), and the application layer, where the physical layer and the medium access control layer are defined by the IEEE 802.15.4 standard [5]. The network layer (NWK) as well as the application layer (APL) are based on the ZigBee standard, which is developed by the ZigBee Alliance [5]. The ZigBee protocol architecture is illustrated in Figure 2.1.

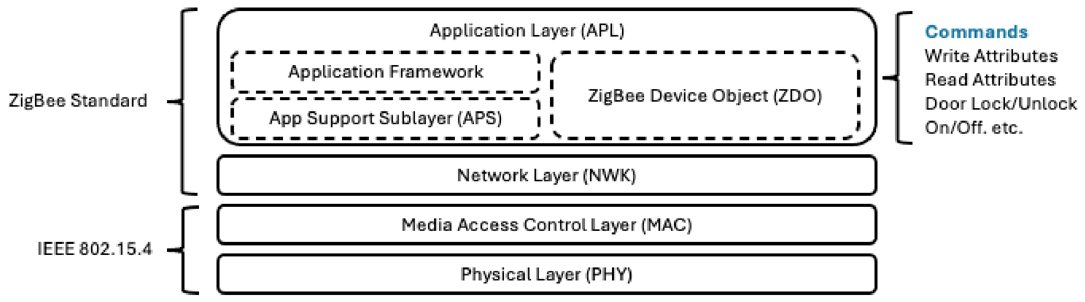


Figure 2.1: ZigBee Protocol Stack based on [7]

The physical layer is closest to the hardware and directly controls and communicates with the radio transceiver [5]. It provides the most basic services such as data interfaces [1] and also selects the channel frequency and makes sure that the channel is not used by other devices on another network [5]. The MAC layer manages the initiation, maintenance, termination, and confirmed transmission and reception of wireless data links between devices [1]. It also provides the interface between the PHY layer and the NWK layer [5]. The NWK layer serves as the intermediary between the MAC and the application layer, managing network formation, routing, and the secure transmission of both outgoing and incoming frames [5] [1]. Routing involves selecting the appropriate path for relaying messages to their destination devices [5]. Meanwhile, the APL layer comprises the Application Support Sublayer (APS), ZigBee Device Object (ZDO), and Application Framework [7]. The APS Sublayer maintains binding tables and address mappings, and the ZDO implements the device into one of three logical roles: ZigBee coordinator (ZC), Zigbee Router (ZR), or ZigBee end-device (ZED) (see section ZigBee Device Types) [7]. The Application Framework provides predefined profiles (e.g., ZigBee Home Automation (ZHA), ZigBee Smart Energy (SE), ZigBee Light Link (ZLL), etc. [8]) along with functional domains known as clusters (e.g., lighting, security, etc.) to ensure interoperability for end manufacturers [7]. Generally, APL commands are categorized as either function-specific (e.g., turn light on/off, door lock/unlock, etc.) or generic, such as Read Attributes (e.g., query a bulb/device for its brightness level), Report Attributes, etc. [7]. The ZigBee Alliance defines clusters, attributes, and commands in the ZigBee Cluster Library (ZCL) specification [9]. The ZCL, therefore, provides a collection of standardized clusters designed for use across various market sectors [10]. Each cluster represents a specific functionality through a set of attributes and commands [10]. Applications can select clusters from the ZCL to incorporate the desired capabilities [10].

ZigBee Device Types

A ZigBee network comprises three main types of devices: the coordinator, end device, and router. The coordinator plays a central role, overseeing the creation, management, and operation of the network [1]. End devices, often sensor nodes, are responsible for monitoring and gathering environmental data [1] or executing actions based on user commands [6]. Routers function as intermediate nodes, facilitating the transmission of data packets across the network [1]. However, in a typical smart home setup, only the coordinator and end devices are commonly utilized [1] as shown in figure 2.2.

ZigBee Light Link

ZLL is a public application profile that has been developed by the ZigBee Alliance to enable consumer lighting solutions and is implemented in the application layer of the ZigBee protocol stack [11]. In contrast to traditional ZigBee networks, a ZLL system does not rely on a ZigBee Coordinator [11]. Instead, network formation and joining are facilitated through a commissioning application called Touchlink, designed to offer consumers a simple and intuitive installation process [11]. The ZLL profile defines standards for controlling various lighting parameters, including on/off states, color hue, saturation,

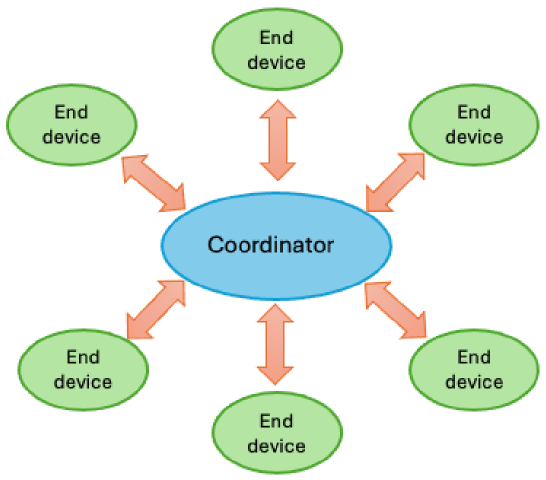


Figure 2.2: Typical ZigBee network topology for a smart home based on [1]

brightness, and color temperature [11]. It enables users to group lightning devices for simultaneous control and configure specific light settings as "scenes" that can easily be recalled [11]. This is implemented by using clusters of Color Control, Level Control, On/Off, Scenes, and Groups [11]. A system can accommodate multiple groups with individual lamps belonging to one or more groups [11]. The brightness and color (or color temperature) may also be collectively adjusted to create scenes tailored to specific moods or activities, such as watching TV, reading, dining, or party [11]. A system can support multiple custom scenes [11]. ZLL also enables home lightning systems to be connected to the Internet, allowing control from any Internet-connected device, such as a PC or a smartphone, from anywhere in the world (e.g., while on vacation abroad) [11]. To facilitate Internet access, the ZLL system must include a control bridge, which serves as a network device connecting the system to the Internet [11]. One such ZLL system is the Philips Hue LED web-enabled system [11].

2.1.2 Philips Hue

Philips Hue is considered the most popular lighting system for homes today [8]. The platform consists of LED light bulbs equipped with a chip that receives commands through the control bridge using the ZigBee communication protocol [12]. With the app users send these commands via the Internet and/or home router to the bridge, which transforms the commands into ZigBee command frames and transfers them to the light bulbs [8]. Its architecture allows for endless connections with other smart home devices [12]. A setup is depicted in figure 2.3. In addition to the official Hue app, hundreds of complementary third-party apps have been developed for Hue since Philips Hue released a designated developer portal with an official API documentation and software development kit (SDK) in March 2013 [12]. One such example is iLightShow which synchs Philips Hue Lights to Spotity or Apple Music [13].

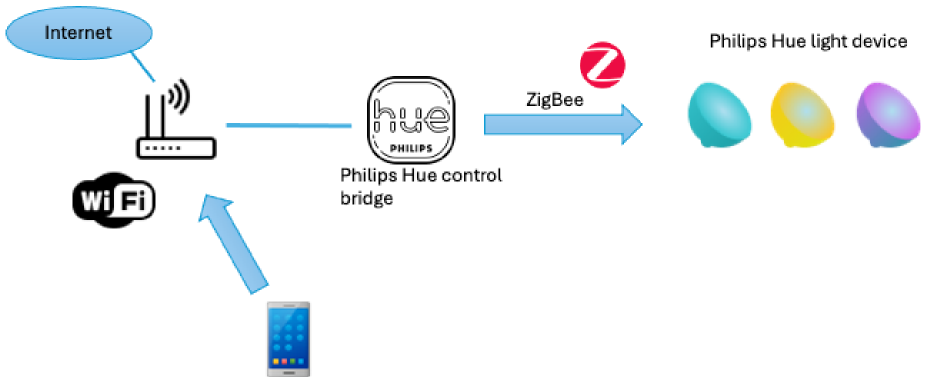


Figure 2.3: Philips Hue System Architecture, adapted from [11]

2.1.3 User Profiling

While the collection of a single piece of user data might seem harmless and not very informative, the aggregation of such data over time allows systems to construct profiles of the customers, potentially revealing sensitive information about the individual [14]. Wang and Loui [14] emphasize that it is this aggregation, rather than isolated data points, that poses privacy risks. To understand how such profiles are constructed and used, this section explores the fundamental concepts of user profiling derived from the comprehensive survey by [15]. Additional sources are cited accordingly.

Advancements in information and communication technologies have driven the development of personalized information systems designed to adapt their functionalities to users' specific interests. One important application of user personalization is the recommender system [16]. For instance, [17] highlights the inefficiencies of search engines that fail to provide relevant information due to factors such as the increasing available information and services in digital format, the variety of user goals, and bad query formulation, which underline the need for adaption, contextualization, and personalization.

To address the issue of personalization, a comprehensive user profile can be built, which is a summary of the user's interests, characteristics, behaviors, and preferences. User profiling, on the other hand, is the system of collecting, organizing, and inferring the user profile information. In this sense, the user profiling system addresses the challenges posed by [17] by constructing and demonstrating tailored information about each user.

Whilst user profiling can be beneficial for tailoring a system or a device to a user and making their experience more personalized, the information contained in a user profile is personal. Hence, there are privacy issues arising [18]. While it can be assumed that an ethical and trustworthy service would only use the collected information in a user profile with the user's explicit consent, there are services which do not care about protecting the user's privacy [18]. The personal information in user profiles can be sold to third parties for profit, which could then be used for commercial or malicious purposes [18]. But even if a service protects the privacy of its users, privacy breaches can still occur [18]. Also the recent increase in intrusive and privacy-invasive advertisements has sparked significant concerns among users and regulatory bodies [19]. Online advertising remains the dominant monetization model on the web [19]. It increased growth rates and revenues tremendously, promoting advancements in user tracking and profiling technologies to serve more relevant ads to the user [19]. Web tracking and user profiling utilize mechanisms to uniquely identify and monitor a user's online behavior over time, such as geolocation, visited pages, search keywords, and social network activity to gain a deeper understanding of the user's intentions and interests [19].

According to [20], to implement a user profile, a user model must first be created. User modeling is a field that focuses on how user information can be acquired and used by automated systems. The data that reflects the user's interests is typically referred to as a user profile [20]. Different user models exist:

- *Behavioral modeling:* This model is based on human behavioral patterns. The data gathered during the interaction between the system and the user is stored, and their actions are analyzed [21]. Based on those analyzed actions, an estimation of the intended action is made [21].
- *Preference modeling:* Based on past interactions and choices that users made this approach deduces user preferences and predicts what a user might prefer in the future [22].
- *Interest modeling:* This model tries to identify and understand topics, subjects or content that attracts the user's attention [22]. By constructing a user interest model, systems can deliver more personalized recommendations, content suggestions, and targeted information [22].
- *Intention modeling:* This model tries to identify the user's goal when interacting with the system [21]. Users can then be grouped into categories based on their intentions (e.g., a group with the intention of buying and a group without the intention of buying a product) [21]. It tries to identify the ultimate reason why a user started interacting with a system and is developed upon the foundation of behavioral and interest modeling [21].

Not only is a user profile used to hold current information about a user and provide them with accurate data. According to [17], it can also be used to predict the user's future intentions and actions.

Further, [23] categorizes user profiles into two types: static and dynamic profiles. The static profiling approach focuses on analyzing a user's consistent and predictable characteristics. The information in a static profile is aggregated and maintained for an extended period of time without undergoing changes or modifications, such as the user's age and gender. The information used to build a static user profile is mainly supplied by the user itself (e.g., by filing online forms/surveys, or during account creation, etc.) [16] [22]. However, users rarely like to disclose all their information or don't disclose it accurately, as they don't want to compromise their privacy, making the static profile less reliable.

The dynamic profiling approach, on the other hand, does not rely on manual input from the user. Instead, the profile is automatically generated and continuously updated by the system based on observed user behaviors and interactions, allowing user attributes and preferences to change over time. According to [16], the profile information in dynamic profiling focuses more on predicting future user behavior or preferences rather than just reflecting their current or past actions.

Considering time in a dynamic user profiling approach can further distinguish between short-term and long-term profiles. Short-term profiles capture the current interests of the user. In contrast, long-term profiles represent more enduring preferences that remain relatively stable and less prone to frequent changes over time [24]. In the following subsection, the difference between short-term and long-term profiles is explored further.

Dynamic profiling: Short-Term and Long-Term Perspectives

The discussion in this subsection on short-term and long-term profiling is informed by [24].

The distinction between short-term and long-term profiles can differ depending on the specific application, domain, and analytical objectives. Long-term profiles offer insights into the user's enduring preferences and behaviors, which allows for personalized recommendations and tailored experiences over long periods of time. [24] mentions the following examples of what a long-term profile can track: changes in a user's career trajectory, evolving hobbies, or shifting lifestyle preferences. Short-term profiles, on the other hand, focus on capturing immediate shifts in a user's interests and activities, supporting real-time adaption and responsiveness. For example, a short-term profile can reflect temporary interests, such as trending topics, current events, or seasonal preferences. The combination of long-term and short-term profiles provides an extensive understanding of the user's dynamic behavior.

An overview of the user profiling taxonomy is depicted in figure 2.4.

User Profiling in IoT

With the rise of IoT, it is necessary to investigate what the data that can be collected from IoT devices reveal about a person since, in many cases, these devices are connected to a person and, therefore, to the person's movements and actions [4]. Yet, many might not be worried about this fact since this generated IoT data can usually not be directly linked to a person as it doesn't contain personal information such as phone numbers, names, or addresses [4]. The authors in [4] mention, though, that there are cases where this data can be linked to an individual. For example, while the data coming from a connected car might not contain personal information, the car is most likely connected to the owner of the car, and this data will probably contain a lot of information about them [4]. In the context of a light bulb, [25] mentions that many end-users might underestimate the accompanying privacy concerns compared to, for instance, a camera with facial recognition. But even with a light bulb, the user's habits can be easily inferred and differentiated between regular versus exceptional patterns [25]. [4] also show in their work that it is possible to identify users with user profiles they created based on observed user behavior and that it is possible to track and follow a person's habits through digital

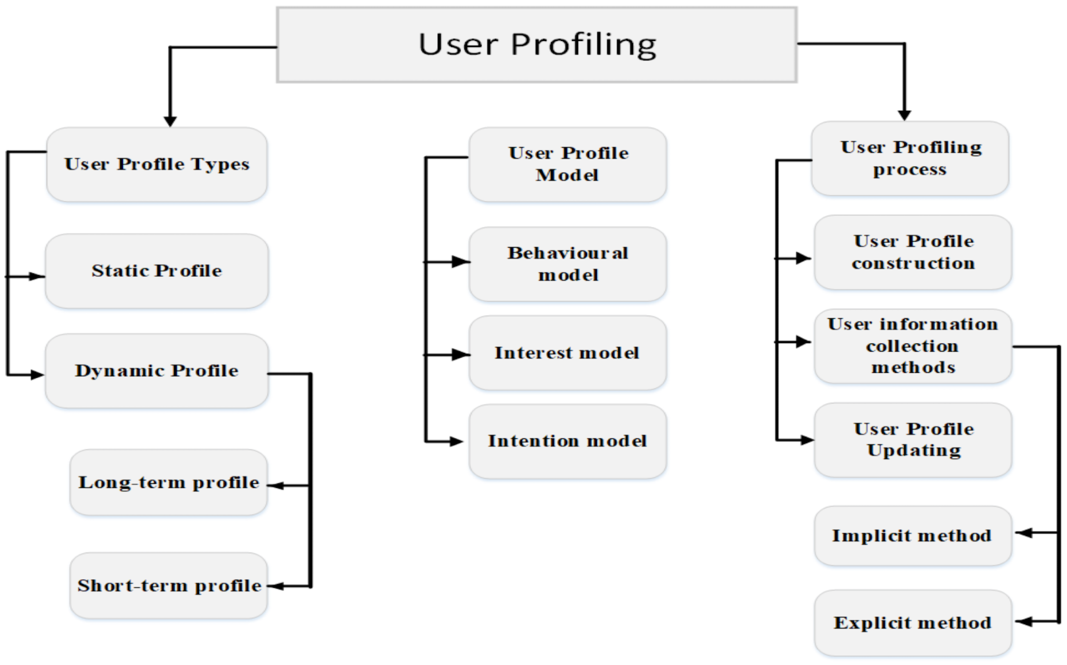


Figure 2.4: User Profiling Taxonomy [15]

patterns. They, therefore, conclude that these profiles, which consist of IoT data, pose a great threat to users' privacy as they can be used to identify individuals or groups of individuals [4].

2.1.4 Data Security and Privacy

The content in this introduction on data security and privacy is primarily based on the work of Bertino [26]. Other sources are cited accordingly.

Today, data have become more critical and relevant than ever. Advancements in technologies and new applications, including sensors, cloud computing, data analytics, social networks, and IoT, have enabled the collection, storage, and processing of vast amounts of data, commonly known as big data. Not only is it possible to store and process these data sets, but it is also possible to extract useful knowledge from this data and predict trends and events with emerging capabilities in the field of data analytics. As we rely more on these technologies, the security and privacy of the data these systems manage become crucial. Not only are single individuals or organizations affected by data misuse, but entire social sectors and critical infrastructures can also be damaged. With data collection and processing having become pervasive, data protection has become much more complex than when data collection and processing was handled solely within individual organizations. In a previous paper by Bertino and Sandhu [27], three basic data security requirements were defined:

- Confidentiality: Referring to data protection from unauthorized accesses.

- Integrity: Referring to data protection from unauthorized modifications.
- Availability: Referring to assuring that data is available to authorized users.

While these three requirements are still critical today, meeting them is much more challenging as data attacks have become more sophisticated and the data attack surface has expanded due to the increasing data collection activities. A new critical requirement has emerged next to the aforementioned three: *privacy*. While data privacy is often seen the same as data confidentiality, there are differences. For data privacy to take effect, data confidentiality must first be assured since privacy can not be guaranteed if the data are not protected against unauthorized access. In addition to that, privacy needs to consider requirements that stem from legal privacy regulations and individual privacy preferences. Bertino mentions the example of one individual who may be fine with sharing their data while another is not. Therefore, systems that manage privacy-sensitive data should also record the privacy preferences of the individuals to whom the data refer, referred to as *data subjects*. Systems should, therefore, not only enforce the access control policies of the organization but also data subject preferences and legal regulations.

Even though the awareness of security aspects of traditional IT is relatively high and well-defined procedures and rules have been defined, the security in IoT is still in its early development [28]. Many manufacturers lack prior experience in cybersecurity and frequently overlook it [28].

But with IoT becoming omnipresent in everyday life, the risk of cyber security attacks on data is increased, and concerns about privacy and safety are raised. Bertino mentions several reasons why IoT systems are at risk, including their lack of well-defined perimeters, their high dynamic, and their continuous change because of mobility. Also, their communication medium and protocols are heterogeneous. One of the greatest risks associated with data collection by IoT devices is that consumers are often unaware of what data is being collected and how it is being used [4]. Even when consumers consent to data collection for a specific application, it remains challenging to predict how the data might be used in the future [4].

The OWASP Internet of Things Project explored vulnerabilities in IoT systems [29]. The lack of encryption or access control of sensitive data anywhere within the ecosystem and users' personal information being stored and used insecurely, improperly, or without permission are listed under the top ten vulnerabilities.

Following these challenges and vulnerabilities, different frameworks have been created, which will be explored in the following subsections.

GDPR: General Data Protection Regulation

The General Data Protection Regulation (GDPR), which was put into effect on May 2018, is a privacy and security law [30]. Although it was developed and passed by the European Union (EU), its scope extends globally, applying to any organization that targets or processes data from individuals in the EU [30]. The GDPR provides detailed definitions for a wide range of legal terms [31]. Below are some definitions listed and summarized that are deemed relevant for this thesis:

- Personal data: Any information relating to an identified or identifiable natural person ('data subject'). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, identification number, location data, etc.
- Processing: Any operation or set of operations that is performed on personal data or on sets of personal data, whether or not by automated means. Examples cited in the text include collection, recording, organization, structuring, storage, adaptation, retrieval, consultation, erasure, or destruction;
- Profiling: Any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person, in particular, to analyze or predict aspects concerning that natural person's performance at work, economic situation, health, personal preferences, interests, reliability, behavior, location or movements;
- Controller: A natural or legal person who determines the purposes and methods of processing personal data (e.g., an owner or employee of an organization that handles data).
- Processor: A natural or legal person, public authority, agency, or other body that processes personal data on behalf of the controller (thus a third party).
- Third party: A natural or legal person, public authority, agency or body other than the data subject, controller, processor and persons who, under the direct authority of the controller or processor, are authorized to process personal data;

The GDPR defines the following data principles that must be adhered to follow the GDPR's rules [30]:

1. Lawfulness, fairness and transparency: The processing must be lawful, fair, and transparent to the data subject.
2. Purpose limitation: The data must be processed for the legitimate purposes specified explicitly to the data subject when the data is collected.
3. Data minimization: Only as much data as absolutely necessary for the purposes specified shall be collected.
4. Accuracy: The personal data must be kept accurate and up to date.
5. Storage limitation: Personally identifying data shall only be stored for as long as necessary for the specified purposes.
6. Integrity and confidentiality: Processing must be done so that it ensures appropriate security, integrity, and confidentiality (e.g., by using encryption).
7. Accountability: The controller is responsible for being able to demonstrate GDPR compliance with all the aforesaid principles.

NIST Privacy Framework

The National Institute of Standards and Technology (NIST) is part of the U.S. Department of Commerce [32]. With the rise of the Internet and information technologies and the accompanying data accumulation, individuals may not understand the consequences of their privacy while interacting with systems, products, and services [32]. Also, organizations are not aware of the full extent of these consequences for individuals, society, or their own enterprise [32]. Therefore, in 2020, NIST published a Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management to facilitate improved privacy engineering practices that align with privacy by design principles and assist organizations in protecting the privacy of individuals [32]. This section is based on this tool [32].

Privacy is a challenging field since the means to achieve it can vary, and important concepts such as human autonomy and dignity are not fixed constructs. They may change depending on the cultural or individual differences which makes it difficult to communicate clearly about privacy risks within and between organizations and with individuals. The NIST Privacy Framework provides a structured approach for organizations of all sizes to any particular technology, sector, law, or jurisdiction to identify, assess, and manage privacy risks. Its purpose is to help organizations to manage privacy risks by

- Considering privacy throughout the design and deployment of systems, products, and services that impact individuals.
- Communicating their privacy practices.
- Promoting collaboration across organizational teams–for instance, among executives, legal departments, and information technology–by developing Profiles, selecting Tiers, and achieving desired outcomes.

The Privacy Framework consists of three key components: Core, Profiles, and Implementation Tiers. Each element reinforces privacy risk management by connecting business objectives, organizational roles, and privacy protection practices.

- The Core represents a collection of privacy protection activities and outcomes designed to facilitate effective communication of prioritized privacy measures across all levels of an organization, from executives to operational staff.
- A Profile represents an organization's current privacy practices or intended outcomes. To create a Profile, the organization can evaluate the activities and outcomes outlined in the Core and determine which are the most important to focus on.
- Implementation Tiers serve as a reference point for assessing how an organization views privacy risks and whether it has adequate processes and resources to address and manage those risks effectively.

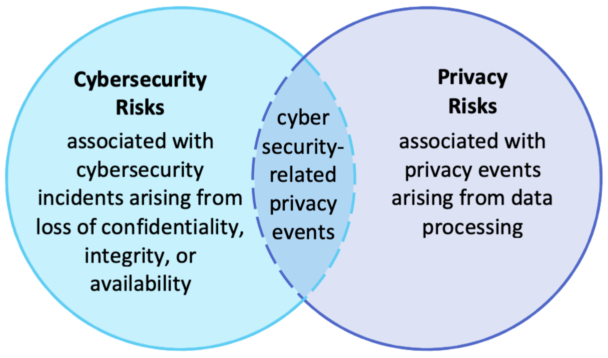

Cybersecurity Risks
associated with cybersecurity incidents arising from loss of confidentiality, integrity, or availability
cyber security-related privacy events
Privacy Risks
associated with privacy events arising from data processing

Figure 2.5: Cybersecurity and Privacy Risk Relationship [32]

In 2014, NIST published a Cybersecurity Framework to help organizations communicate and manage cybersecurity risks. While managing cybersecurity risks might reduce privacy risks, it is not sufficient, as privacy risks can also arise unrelated to cybersecurity incidents. In Figure 2.5, the difference between cybersecurity risks and privacy risks is depicted.

The following lists the functions of the Core to give examples of the privacy protection activities and outcomes:

1. Identify-P (ID-P): Emphasizes developing the organizational understanding to manage privacy risk for individuals arising from data processing (includes, for example, ID.IM-P1: Systems/products/services that process data are inventioried).

2. GOVERN-P (GV-P): Emphasizes developing and implementing the organizational governance structure to enable an ongoing understanding of the organization's risk management priorities that are informed by privacy risk (includes, for example, GV.PO-P4: Privacy roles and responsibilities are coordinated and aligned with third-party stakeholders).

3. CONTROL-P (CTP): Emphasizes developing and implementing appropriate activities to enable organizations or individuals to manage data with sufficient granularity to manage privacy risks (includes, for example, CT.DM-P3: Data elements can be accessed for alteration or CT.DM-P5: Data are destroyed according to policy).

4. COMMUNICATE-P (CM-P): Emphasizes on developing and implementing appropriate activities to enable organizations and individuals to have a reliable understanding and engage in a dialogue about how data are processed and associated privacy risks (includes, for instance, CM.AW-P3: System/product/service design enables data processing visibility and CM.AW-P5: Data corrections or deletions can be communicated to individuals or organizations (e.g., data sources) in the data processing ecosystem).

5. PROTECT-P (PR-P): Emphasizes developing and implementing appropriate data processing safeguards (includes, for instance, PR.DS-P5: Protections against data leaks are implemented).

Privacy in the Philips Hue system

[33] performed a security analysis of the touchlink commissioning in ZigBee 3.0 and showed that it is inherently insecure. The touchlink communication relies on inter-PAN frames that lack both encryption and authentication [33]. Additionally, the network key is transmitted to a joining device by using only a global master key, known as the touchlink preconfigured link key [33]. This key, originally distributed to touchlink-enabled product manufacturers under a non-disclosure agreement (NDA), was leaked in March 2015 [33]. Because of the backward compatibility demands towards legacy ZigBee Light Link products the key can not be renewed [33]. Thus, the authors warn about adopting touchlink commissioning in all future ZigBee 3.0 devices and recommend a different commissioning combination.

In 2023, ZigBee PRO 2023 was released, promising enhanced security features [34]. New mechanisms have been implemented for protecting the network during the onboarding and operation of devices to address modern-day security threats [34]. However, no public information could be found on whether ZigBee Light Link has been implemented into ZigBee PRO 2023 as of today (January 2025).

2.2 Related Work

Different research has investigated potential security and privacy issues related to Philips Hue. [33] performed a security analysis of the touchlink commissioning option in ZigBee 3.0. They show that active and passive attacks are possible and that attackers can hinder the availability of touchlink-enabled devices and gain control over all nodes in the network. [8] conducted a comprehensive security analysis of the most popular ZigBee Light Link (ZLL)-based connected lighting systems, including Philips Hue, Osram Lightify, and GE Link. They revealed that the ZLL standard is inherently insecure, allowing attackers to gain unauthorized control over these systems. For instance, they forced bulbs to accept a new network key, which allowed them to send commands to the bulbs to turn them on or off and change their color, removed bulbs from the legitimate networks and caused the bulbs to blink for several hours, whilst the legitimate user couldn't stop it unless they disconnected the bulbs physically. They also showed that they were able to extend the typical wireless configuration range from 2 meters up to 36 meters, making these systems prone to remote attacks. In [35] the authors developed a worm infection using the Philips Hue smart lamps as a platform. The worm propagates by jumping from one neighboring lamp to another, using solely their built-in ZigBee wireless connections and close physical proximity. [25] studied how using different techniques to control a Philips Hue smart bulb could immensely change the security and privacy risks.

Transitioning to the broader spectrum of sniffing and potentially revealing user activities and behaviors, several research has been conducted. In [1] ZPA was developed, a privacy detection system for smart homes that examines the encrypted ZigBee traffic to identify potential information leaks. Their analysis of 20 devices from five manufacturers demonstrated that, despite encryption, the system was able to recognize the device type with

93% accuracy and the specific device states with 98% accuracy. They show that encrypted ZigBee traffic can expose user activity information. [36] proposes a tool called Stealth-Profiler, which is a proof-of-concept tool designed to identify devices and events within a Philips Hue network without decrypting network traffic. They show that passive and active tracking are possible and that possessing a single network key allows for potential decryption in any other Philips Hue network, regardless of their network keys.

[37] proposed a method to eavesdrop on different kinds of user privacy issues by analyzing the wireless context in a smart home, including a Philips Hue system. They investigated detected IoT events and inferred the user's activities and even the user's mood. They conclude that by analyzing the wireless context, they can infer the user's living habits, routines, and installed IoT applications. They emphasize that they don't try to decrypt the data but only work with the available non-encrypted traffic data. Thus, while they might not know the color and brightness level the user sets, they do know the user has changed the color and brightness level. They use this information to infer the user's mood, as they state a person with stable emotions might not change their light settings for some time, whereas a person suffering from insomnia or irritability might more frequently change it in a short period of time. Further, [38] have also developed a strategy that a passive network observer could apply to infer consumer behavior from rates of IoT traffic, even when the traffic is encrypted. They examined a Sense Sleep monitor, a Nest Cam Indoor security camera, a Belkin WeMo switch, and an Amazon Echo to infer user behavior, such as sleeping patterns. They also emphasize that while the questions asked to Amazon Echo are encrypted and thus not understandable to the network observer, simply knowing the times of day when customers interact with a particular service is a privacy concern. They conclude that encryption alone is not sufficient to provide adequate privacy protection for smart homes.

In [39], a machine learning approach was developed for detecting and identifying the types of IoT devices, their states, and ongoing user activities by only passively sniffing the network traffic from smart home devices and sensors. They showed that their attack works on both encrypted and decrypted communications. In [40] IoTMosaic was developed, a system to profile diverse user activities with distinct IoT device event sequences based on smart home network traffic. They analyzed thousands of user activities in the smart home environment over a period of two months and showed that their algorithm can infer user activities from IoT network data. In [41], the authors analyzed network traffic generated by a smart thermostat and a smart smoke and carbon dioxide detector by Nest Labs, out of which the first can learn a user's schedule and preferences to adjust the temperature of the home. They show that potentially sensitive information about the state of the smart home can be exposed by traffic analysis and analyzed to build user profiles. In [42], passive attacks against wireless home automation systems (HomeMatic) were conducted. The authors showed that these systems provide no privacy. Information about the users' habits was leaked, and the user's presence was exposed, potentially aiding the abuse of burglary. In [43], Chatterhub was introduced, a system that employs passive eavesdropping on encrypted network traffic to infer devices and device states with machine learning techniques that could potentially reveal the household's daily routine.

Comparison of Related Work					
Paper	Philips Hue	Privacy Concerns	Security Concerns	Passive Network Analysis	User Profiling
[33]	●	○	●	◐	○
[8]	●	○	●	○	○
[35]	●	○	●	○	○
[25]	●	●	○	●	○
[1]	○	●	●	●	○
[36]	●	●	●	●	◐
[37]	◐	●	○	●	●
[38]	○	●	○	●	●
[39]	◐	●	◐	●	●
[40]	○	●	◐	●	○
[41]	○	●	○	●	○
[42]	○	●	◐	●	●
[43]	○	●	○	●	◐

○: Not investigated ◐: Partially investigated ●: Fully investigated

While [37] and [39] address all the points listed, their primary focus is not on the Philips Hue system. Instead, they analyzed broader smart home environments in which Philips Hue devices were included as part of the ecosystem. In contrast, this work will focus solely on the Philips Hue system, enabling a more detailed and in-depth analysis. Also, while [36] addresses all the identified points, user profiling is not specifically implemented. Building on [6] and [36], this work will additionally implement a user profiling system.

Chapter 3

Design

3.1 Building and Modeling User Profiles

This section builds on the work of [22] and follows it to work out the design of the user profiling application. Other sources are cited accordingly.

3.1.1 User Profile Construction

An important aspect of user modeling involves constructing profiles, which can be categorized into individual and group profiling. Individual profiling is based on data linked to a single user and is the predominant approach in most studies in this research area. Group profiling, on the other hand, focuses on a collection of users who share common interests, goals, or preferences and creates a partially complete profile by identifying commonalities across multiple users.

For this thesis, an individual profile is considered the most appropriate as the goal is not to find similarities across different users but rather to discover the particular routine and preferences of a single user or household.

3.1.2 Data Collection Approach

As discussed in the section User Profiling, a user model must first be created to define how the system acquires and uses user information. Four different user models were presented: behavioral -, preference-, interest -, and intention modeling. All four approaches are part of the implicit user profiling data collection approach, which refers to the passive collection and analysis of dynamic user data without requiring direct input from the user. Next to the implicit user profiling approach, there exists also the explicit-, the pseudo-explicit-, and the hybrid user profiling appraoch. The explicit user profiling approach relies on direct user input (e.g., through the completion of online forms, questionnaires, or through the provision of ratings and preferences). The pseudo-explicit user profiling approach uses data that the user has shared willingly at some point for different reasons (e.g., using a travel platform or setting up a social network account). However, this gathered data was not originally provided by the user for the intent of profiling but rather shared for other purposes. The hybrid user profiling method combines the implicit and explicit user profiling approaches and uses the benefits of both. As this thesis builds on the work of Datsomor [6], who created an application for passively sniffing ZigBee network data, the implicit user profiling approach will be applied.

Since this thesis's goal is to investigate the data that is created as the user interacts with the system and tries to identify patterns in the user's behavior, the behavioral model was deemed appropriate. The preference model is considered important as well for this thesis since it might be possible to derive what settings of the Philips Hue system the user likes or dislikes (e.g., the color of the lamps), thus revealing more about their personal preferences.

3.1.3 User Profile Representation

The user profile representation describes how information in user profiles is structured, stored and displayed in a system or application. Its purpose is to create an accurate description of the user, allowing the system to make informed decisions and predictions about their interactions and preferences. There exist different representation approaches:

- *Term-based (Vector-space model):* User interests are represented through keyword-based or vector-based user models, where terms are weighted with vectors. For example, a user's interest keywords, which are extracted from visited documents during browsing, can be represented by a single vector comprising all interests or by multiple vectors that illustrate the user's interest in various domains. However, this model struggles with polysemy (words having multiple meanings) and synonymy.

- *Semantic-based (Ontology-based):* This model mitigates polysemy and synonymy by mapping keywords into weighted concepts within a semantic network to help the system better understand the actual meaning behind the words. However, this can introduce complexities as it requires a way to connect words to their correct meanings, for example, with pre-built word databases, machine learning, or manual input.

- *Concept-based (Hierarchy-based):* Represents user behaviors and interests as a structured hierarchy of concepts and relationships [22]. Each concept is assigned a weight, indicating the user's level of interest in that topic [44].

- *Graph- and Knowledge graph-based:* Represents user information as a network of graphs that can effectively capture complex relationships, dependencies, and interactions among various elements in a user profile. Knowledge graphs, a specific form of graph-based representation, facilitate an understanding of individual profiles, which allows the system to adapt dynamically to a user's preferences and needs.

- *Universal user representation:* The goal is to build a broad user model that captures the essential characteristics and behaviors of a user. The universal user representation prioritizes flexibility, allowing it to accommodate diverse users and their needs while remaining applicable across different contexts and user groups.
- *Holistic user model:* Is about creating a detailed and comprehensive profile that considers all relevant aspects of a user's interaction with a system. This includes not only their actions but also their expectations, preferences, and limitations.

For this thesis, a hierarchical concept-based user profile is deemed the most appropriate as a structured and easily understandable classification of user behavior and routines is needed. Other representations were rejected due to the following reasons:

- *Term-based and Semantic-based models*: These are designed for text-based analysis where the user's interests are inferred from document content or search queries. Since this thesis works with event logs it was not considered suitable for further exploration.
- *Graph-based representation*: While this approach was deemed interesting it was considered to introduce too much complexity for the system that is required to be built. As this thesis must categorize user behavior patterns rather than model complex dependencies and relationships this representation was discarded.
- *Universal and holistic model*: These representations are designed for a broad user understanding. But this thesis's goal is to capture only routines and potential preferences of the user, and this representation was therefore discarded.

Concept-based user profile

Most concept-based approaches identify user's main interests by analyzing the content of the documents they have viewed and their search history [44]. For instance, [44] developed a concept-based profiling system by extracting meaningful concepts from web snippets returned by search engines. They identified frequently occurring keywords in search results to determine which concepts were relevant to the user and established relationships between these concepts by analyzing how often they coexist in documents. Similar search queries were clustered, and a personalized concept-based user profile was built. However, this text-based approach does not directly apply to this thesis task, as the user profiling system that is aimed to be built is based on event data rather than textual data, but no example could be found on how to implement a concept-based user profile for sequential event-based data. Therefore, this thesis draws inspiration from the concept-based approach by adopting a hierarchical structure, but redefines how concepts are determined and weighted:

- Instead of dynamically extracting concepts from user data, this thesis predefines a fixed hierarchy of patterns and sub-patterns that are potentially valuable for an attacker (e.g., burglar or stalker).

- Instead of assigning weights based on user interest in a topic, weights are replaced with probability values, indicating the likelihood of a pattern occurring at a specific time.

The hierarchical structure consists of:

1. **Days of the week** (e.g., Monday, Tuesday, etc.).
2. **Patterns** (e.g., Daily Routine, Night Routine).
3. **Sub-patterns** (e.g., Wake-up Pattern, Leaving Home Pattern, Midnight Activity Pattern, etc.)
4. **Time intervals** within which events most frequently occur.

This hierarchy is illustrated in Table 3.1 and can also be visualized as a tree-like structure (Figure 3.1), where the day of the week is the root node, and patterns and sub-patterns are its children.

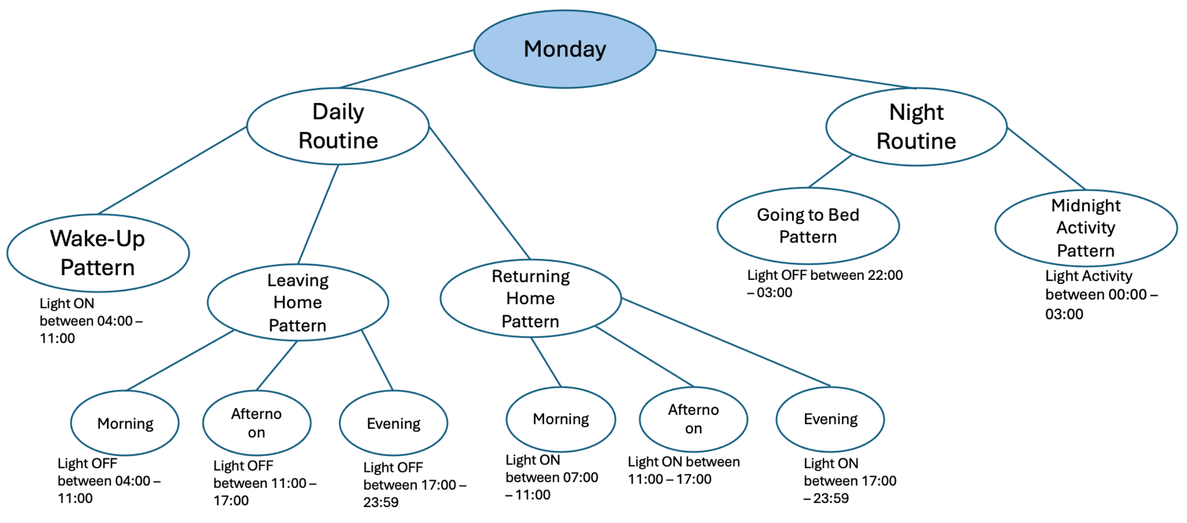

Monday
Daily Routine
Night Routine
Wake-Up Pattern
Light ON between 04:00 – 11:00
Leaving Home Pattern
Returning Home Pattern
Going to Bed Pattern
Light OFF between 22:00 – 03:00
Midnight Activity Pattern
Light Activity between 00:00 – 03:00
Morning
Light OFF between 04:00 – 11:00
Afterno on
Light OFF between 11:00 – 17:00
Evening
Light OFF between 17:00 – 23:59
Morning
Light ON between 07:00 – 11:00
Afterno on
Light ON between 11:00 – 17:00
Evening
Light ON between 17:00 – 23:59

Figure 3.1: Example Tree Structure with the weekday as the root

Unlike the traditional concept-based profiling, where topics and concepts are dynamically derived from text, this system analyzes recurring event patterns within predefined time windows.

- It is assumed that users don't perform activities at exactly the same time every day of the week. Instead of tracking an exact time (e.g., "Leaves home at 07:32"), time intervals of 15 minutes are used (e.g., "Leaves home between 7:30-07:45").
- Each detected event is assigned to the corresponding 15-minute intervals, and probabilities are calculated based on the frequency of events in each interval.
- If a user, for example, left home 10 times on Mondays and 7 times this happened between 07:30-07:45, this interval would be assigned a probability of 70%.

Days of the Week	Pattern	Sub-pattern	Time Interval Considered	Command Considered	Most Frequent Command Detected	Alternative Events Detected
Monday	Daily routine	Wake-up pattern	04:00-10:00	Lights ON	06:00-06:15 (67%)	06:30-06:45 (33%)
		Leaving Home	04:00-11:00	Lights OFF	07:30-07:45 (60%)	07:15-07:30 (25%), 08:00-08:15 (15%)
			11:00-17:00	Lights OFF	14:00-14:15 (30%)	13:45-14:00 (20%), No event detected (50%)
			17:00-23:59	Lights OFF	19:30-19:45 (50%)	20:00-20:15 (30%), 18:45-19:00 (20%)
		Returning Home	07:00-11:00	Lights ON	08:30-08:45 (40%)	09:00-09:15 (35%), No event detected (25%)
			11:00-17:00	Lights ON	17:45-18:00 (78%)	18:15-18:30 (22%)
			17:00-23:59	Lights ON	21:00-21:15 (50%)	20:45-21:00 (30%), No event detected (20%)
	Night routine	Going to bed pattern	22:00-03:00	Lights OFF	23:00-23:15 (90%)	22:45-23:00 (10%)
		Midnight Activity Pattern	00:00-03:00	Lights ON	02:00-02:15 (20%)	No event detected (80%)
Tuesday	Daily routine	-	-	-	-	-
	Night routine	-	-	-	-	-
etc.	Daily routine	-	-	-	-	-
	Night routine	-	-	-	-	-

Table 3.1: Hypothetical example of user routines based on event logs, illustrating potential analysis outcomes

3.1.4 Modeling Techniques

After organizing the information, the next step is to choose suitable techniques to develop a user profile. This stage focuses on building a computational model using the extracted features and enables the system to predict the user's needs or preferences. There exist many different modeling techniques, as illustrated in Figure 3.2, but detailing all of them would go beyond the scope of this design section. Therefore, only the most relevant approaches will be compared and evaluated in Table 3.2.

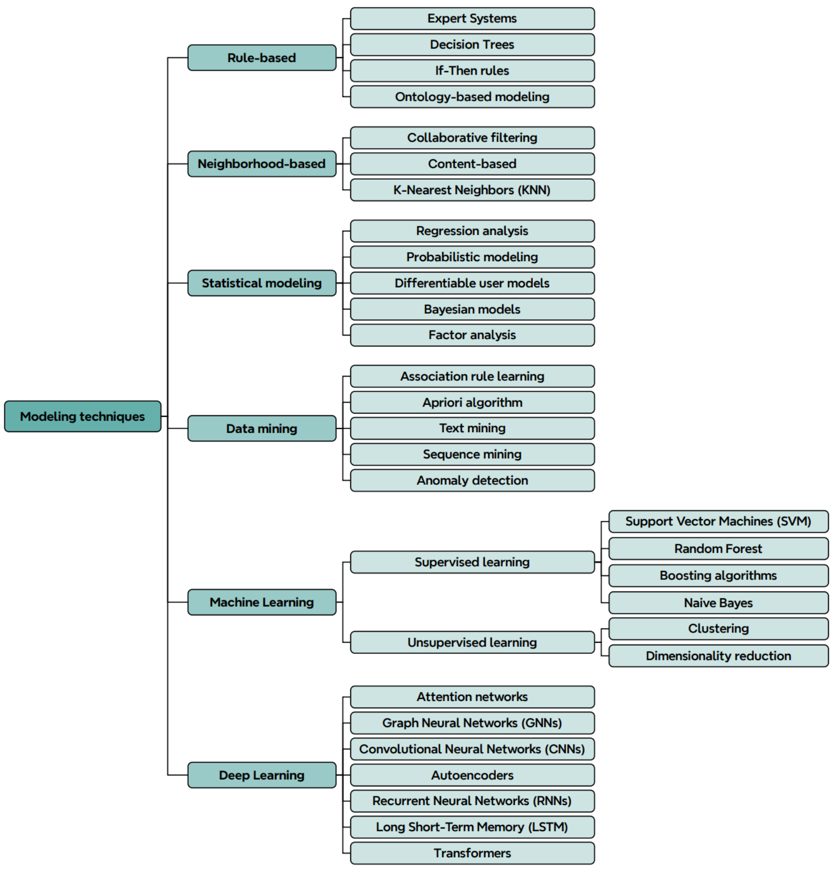


Figure 3.2: Taxonomy of the Modeling Techniques [22]

The following comparison and evaluation of different user modeling techniques are informed by the work of [22]:

- **The neighborhood-based** approach is not investigated further, as it relies on the principle that users with similar characteristics share common preferences. This method involves comparing a user's behavior or attributes to a group of similar users (a "neighborhood") to make predictions or recommendations. Since this thesis aims to analyze individual routines rather than compare behaviors across multiple users, this approach is not deemed suitable.
- **Statistical modeling** of user behavior involves applying statistical techniques to analyze patterns in user data, uncover relationships, and generate predictions. However, since this thesis focuses on identifying predefined user routines based on event-

driven data rather than modeling statistical dependencies or making probabilistic inferences, statistical modeling is not considered the most suitable approach.

- **Machine learning** involves a range of computational techniques that enable systems to automatically identify patterns and relationships and make predictions based on user data. This approach is considered to introduce a level of complexity that exceeds the requirements of this thesis, where predefined patterns and rule-based detection are prioritized.

- **Deep learning** uses multiple layers of deep neural networks to identify complex patterns and representations within user data. It is particularly effective in processing large and unstructured datasets, making it suitable for tasks such as predicting user behavior and generating personalized recommendations. However, due to the structured nature of the predefined user behavior patterns in this thesis and the absence of a need for complex feature extraction, deep learning is not pursued further.

3.2 Working with Multiple Sniffing Sessions

Since continuous real-time monitoring of the ZigBee network is not feasible, as it would require sniffing the whole day during several days of the week, the decision was made to conduct the sniffing sessions periodically to capture event logs in different timeframes. Each session will generate a CSV file containing timestamped events such as lights turning ON/OFF or adjustments in color or saturation.

To enable long-term analysis and cumulative user profiling, these individual CSV files will be incrementally merged into a structured JSON file, which will serve as the central database for all the recorded events. The following advantages are hoped to be achieved by following this approach:

1. As JSON allows for structured storage, maintaining the chronological order of events across multiple sessions should be possible.

2. Instead of processing multiple CSV files separately, the system can work with a consolidated JSON file, and event detection and pattern analysis might be simplified.

Modeling Technique	**Pros**	**Cons**	**Evaluation**
Data Mining (Sequence Mining	Discovers patterns, associations, and trends within large data sets. Sequence mining can identify sequences of events and identify behavior patterns over time.	Many sequence mining algorithms are designed to discover patterns based on the relative ordering of events rather than their occurrence at specific time points [45]. [45] state that, for example, only a few algorithms could define that a given sequence occurs on Mondays but not on other days. Depending on the chosen algorithm, it is potentially complex to implement.	Sequence mining might be able to detect a recurring pattern that, for example, a Lights Off command at 7:30 AM usually follows a Lights On command at 8:00 AM. But since this thesis would like to work with pre-defined hierarchies and concepts, it is unclear how the detected patterns could potentially align with them.
Rule-Based (If-Then Rules)	In rule-based user modeling, a set of pre-defined rules or conditions that determine user behavior or preferences is created. With If-Then rules, specific conditions (if) are defined to determine specific actions or conclusions (then). In user modeling, these rules are applied to make decisions or predictions based on observed user behavior or input features.	Limited flexibility, as it relies on pre-defined rules and may struggle to adapt to unseen behaviors.	Should integrate well with the pre-defined patterns in the hierarchy-based representation, as it allows explicit categorization of detected behaviors.

Table 3.2: Comparison Sequence Mining and If-Then Rules Modeling Techniques

Chapter 4

Implementation

4.1 Overview

The implemented application uses and extends the previously developed prototype by [6], which is capable of sniffing Philips Hue traffic and logging events such as lights being turned on/off, color changes, and saturation adjustments. This work builds upon the base prototype by introducing a structured and rule-based behavior detection. The core logic processes raw device logs, detects recurring behavior patterns such as waking up, leaving or returning home, going to bed, or midnight activity, and stores these insights into a dedicated user profile. The processed data is visualized through a custom web-based dashboard built using Streamlit to provide a user-friendly overview of behavior trends across multiple sessions and days. The system was later extended to include outlier detection and long-term habit tracking functionality. A broad overview of the system architecture is depicted in Figure 4.1 and presented in detail in the following subsections of the implementation chapter.

4.2 Base Prototype

The implementation developed in this thesis builds upon a ZigBee sniffing prototype previously created by [6], which will be summarized shortly to provide a better overview of how the whole application functions. This earlier system focused on passively capturing Philips Hue network traffic using the ZigBee protocol, processing and analyzing the data to identify core commands such as light on/off, color control, and saturation changes. The system allows for both live tracking and passive analysis modes. While the prototype was initially functional, modifications were required to make the system operational again and extend its capabilities with behavior analysis and user profiling. The network is sniffed with a Nordic Semiconductor nRF52840 Development Kit with the nRF Sniffer for 802.15.4 firmware flashed onto it. Wireshark including Tshark is used for packet capture and analysis.

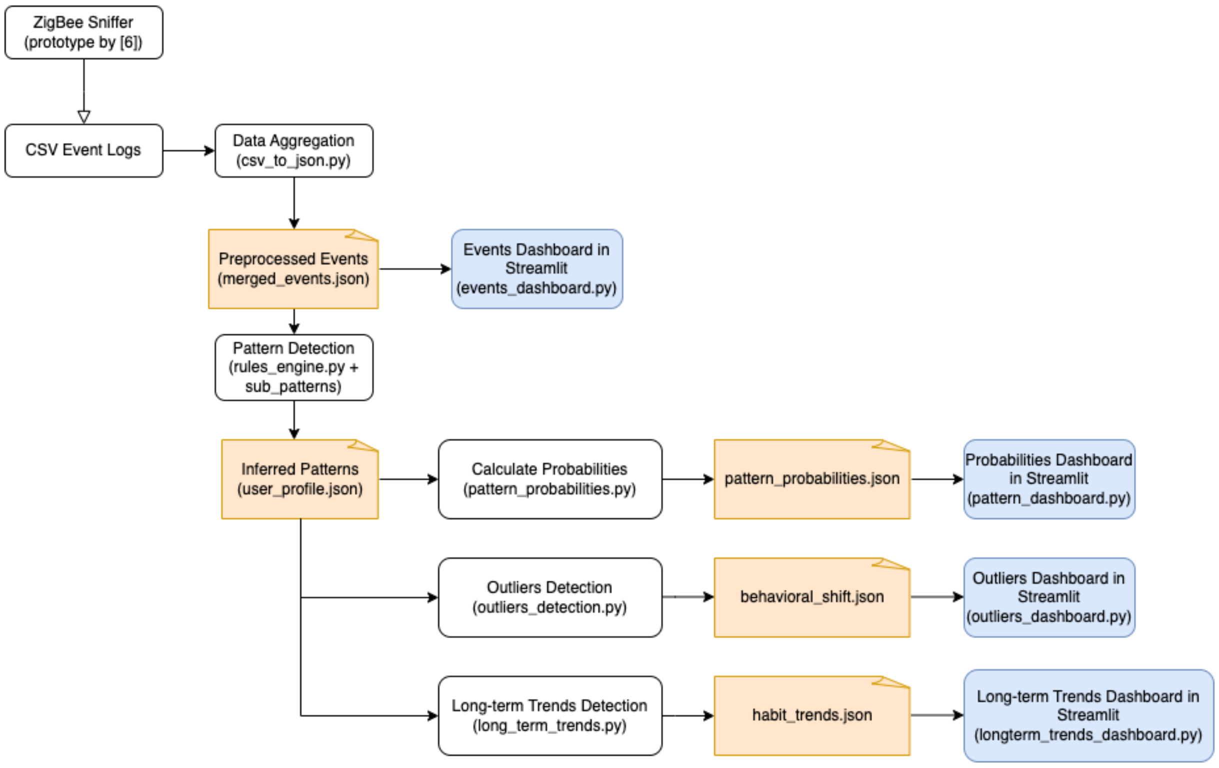

ZigBee Sniffer (prototype by [6])
CSV Event Logs
Data Aggregation (csv_to_json.py)
Preprocessed Events (merged_events.json)
Events Dashboard in Streamlit (events_dashboard.py)
Pattern Detection (rules_engine.py + sub_patterns)
Inferred Patterns (user_profile.json)
Calculate Probabilities (pattern_probabilities.py)
pattern_probabilities.json
Probabilities Dashboard in Streamlit (pattern_dashboard.py)
Outliers Detection (outliers_detection.py)
behavioral_shift.json
Outliers Dashboard in Streamlit (outliers_dashboard.py)
Long-term Trends Detection (long_term_trends.py)
habit_trends.json
Long-term Trends Dashboard in Streamlit (longterm_trends_dashboard.py)

Figure 4.1: System architecture

4.2.1 Core Architecture and Purpose

The main objective of [6] was to extract and analyze ZigBee packet commands, perform device identification, and detect known event types within the network. All components were orchestrated through a central run.py script (Runner), which initiated either passive or live capture, processed the packets, filtered out irrelevant ones, and finally ran inference logic on the event types. The following scripts were included:

- **Command Extractor**: Extracts commands and data lengths from a decrypted and encrypted JSON file, which originate both from the same network capture. The decrypted JSON file is decrypted with the network key. A final data_command CSV file gets produced which maps command and frame lengths to their corresponding data lengths.
- **Packet Extractor**: Converts the captured .pcapng files into a structured CSV format to enable further analysis.
- **Filter**: Outputs a filtered CSV file that contains only ZHA packets to smooth the analysis.
- **Identifier**: Distinguishes between ZigBee device types (e.g., coordinators, routers, end devices) based on packet patterns like route records and read attribute responses.
- **Analyzer**: Applies rules to infer specific user commands (e.g., light ON/OFF) by matching packet signatures and sequencing.

- **Tracker and Runner**: Manage live tracking sessions and the sequential execution of all required scripts, ensuring smooth automation.

4.2.2 Challenges and Functionality

During the execution of the application, it continuously failed and crashed while running the identifier.py script because no coordinator or end device was found during the analysis of new sniffing sessions. It was later realized that the packets' frame length and data length in new sniffing sessions didn't match the command_data CSV output file anymore, which was needed to identify the devices inside the Identifier. Thus, the two JSON files to build the data_command CSV file had to be exchanged with an up-to-date sniffing session. Therefore, two new JSON files were required, one encrypted and one decrypted with the network key. [6] and [1] used a simple process to retrieve the network key during packet capture; a new light device must be added to the Philips Hue network during sniffing in order to obtain the packet containing the network key, which is needed to decrypt the commands. Even after numerous attempts, the packet containing the network key was found, but the information visible in the packet differed largely from [6] and [1]. It was later realized that adding the default global link key into the Wireshark settings, as explained by [1], didn't make a difference in how the data in Wireshark was displayed. The two pre-configured keys required according to the Nordic Semiconductor documentation [46] on how to configure Wireshark for Zigbee didn't decrypt the traffic enough to derive the network key from. After this realization and a brief web search, two other pre-configured keys were discovered in a blog post [47] and tested. The network key was finally visible in Wireshark when adding a new light bulb during sniffing. With the network key available, a sniffing session was conducted and exported in JSON once with the network key decrypted and once without the decryption. These files replaced the old JSON files, and a new data_command CSV file with the new data and frame lengths was produced every time the application was run. The commands Saturation Control (as broadcast message) and Light ON and OFF (also as broadcast messages) were finally derived by running the application over recorded sniffing sessions. Due to time constraints, it was decided to continue with the development of the user profiling system instead of continuing to try to fix the decryption of the color control command and to get back to it after the user profiling system has been built and time remains.

The application built by [6] initially provided a terminal output with information on what date and time the user executed these commands. A small modification in the analyzer script allowed a CSV file to be created that contained the information also visible in the terminal command. This file can be found in the CSV folder of the application called events_logs_session_id, with the session ID being the date and time of the first event found in the logs, e.g., events_logs_20250404_081122.csv for a file where the first event listed is "Apr 4, 2025 08:11:22.634892000 CEST",Turned Light ON (Broadcast),0xfffd".

4.3 User Profiling System Extension

Building upon the functional prototype to infer the ZigBee commands, this section outlines the implementation details of the user profiling application that was implemented and capable of recognizing recurring behavior patterns in the usage of Philips Hue Light bulbs. The system processes sequences of Philips Hue ZigBee commands and infers behavioral routines such as wake-up times, leaving and returning home, bedtime, and midnight activity.

4.3.1 System Architecture Overview

CSV to JSON Script

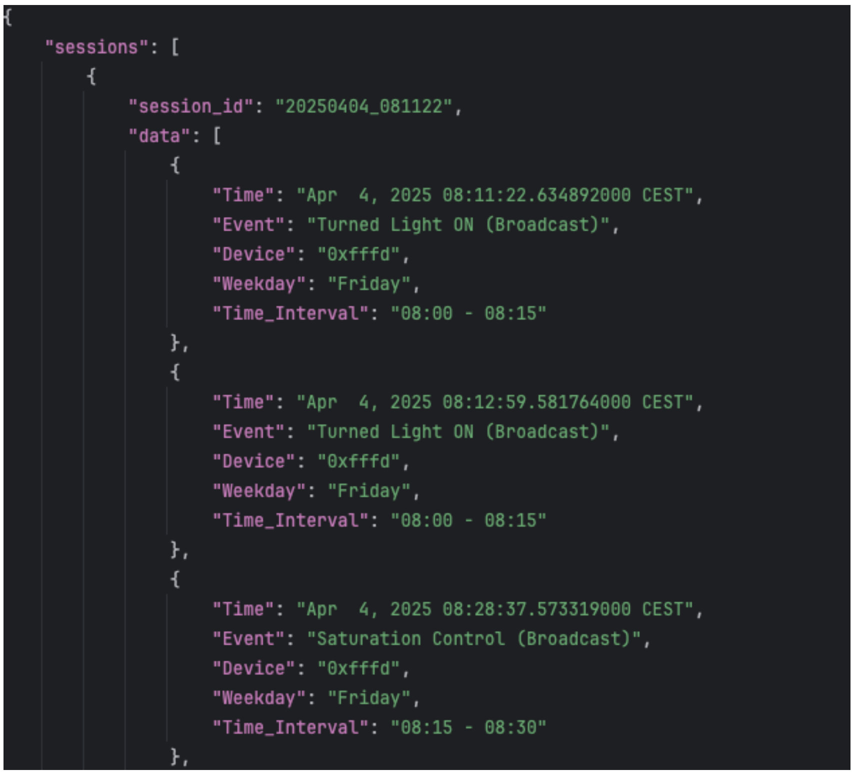
In order to support long-term behavioral analysis across multiple sniffing sessions, the application first consolidates all captured session logs into a centralized JSON file. This is handled by the csv_to_json.py script, which reads each newly generated events_logs_session_id.csv file, enriches the event records with metadata such as the day of the week and a derived 15-minute time interval, and stores them under their respective session in the merged_events.json file. Each session is uniquely identified by the session ID (based on the capture timestamp) to avoid duplicates. Sessions that have already been merged are automatically skipped. An excerpt of this merged structure is shown in Figure 4.2. If no merged_events.json file exists, running the script will create a new one with the current events_logs_session_id as the first input.

Rules Engine Script

Once all raw event logs are consolidated, the system proceeds to detect behavioral patterns using a dedicated rules engine (rules_engine.py). This script applies a set of if-then rules for recognizing high level behaviors such as:

- **Wake-Up Pattern:** Detected based on morning Light ON events.
- **Leaving Home / Returning Home:** Based on temporal patterns in light activity or saturation control.
- **Going to Bed Pattern:** Identified by the absence of activity following a Light OFF command in the late evening.
- **Midnight Activity:** Characterized by brief light activity during night hours.

Each pattern is defined as an individual detection function, which will be presented in more detail in the following sections, and the rules engine coordinates their execution. In Listing 4.1 the main detection flow is visible:



```
{
    "sessions": [
        {
            "session_id": "20250404_081122",
            "data": [
                {
                    "Time": "Apr  4, 2025 08:11:22.634892000 CEST",
                    "Event": "Turned Light ON (Broadcast)",
                    "Device": "0xfffd",
                    "Weekday": "Friday",
                    "Time_Interval": "08:00 - 08:15"
                },
                {
                    "Time": "Apr  4, 2025 08:12:59.581764000 CEST",
                    "Event": "Turned Light ON (Broadcast)",
                    "Device": "0xfffd",
                    "Weekday": "Friday",
                    "Time_Interval": "08:00 - 08:15"
                },
                {
                    "Time": "Apr  4, 2025 08:28:37.573319000 CEST",
                    "Event": "Saturation Control (Broadcast)",
                    "Device": "0xfffd",
                    "Weekday": "Friday",
                    "Time_Interval": "08:15 - 08:30"
                },
```

Figure 4.2: Excerpt of events logged in the merged_events.json file grouped in their corresponding session ID

```
def detect_patterns(json_data):
""" Process sessions and apply pattern rules. """
all_detected_patterns = []
all_events = []

for session in json_data["sessions"]:
    session_id = session["session_id"]
    for event in session["data"]:
        event["Session"] = session_id
        all_events.append(event)

# Sort all events chronologically
all_events = sorted(all_events, key=lambda e: pd.to_datetime(e["Time
    "].replace(" CEST", "").replace(" CET", ""),
                                                                errors=
                                                                    "
                                                                    coerce
                                                                    "))

used_timestamps = set()

for pattern_name, rule_func in PATTERNS.items():
    detected = rule_func(all_events)

    for match in detected:
        if match["Time"] in used_timestamps:
            continue # skip already 'claimed' events
            used_timestamps.add(match["Time"])
            all_detected_patterns.append(match)

    return all_detected_patterns
```

Listing 4.1: Pattern Detection in Rules Engine

The Rules Engine, therefore, takes the merged_events.json file as input, detects the patterns based on the set of rules defined for each pattern, and outputs a structured list of detected patterns, which is stored in a separate user_profile.json file. Events that have already been claimed by one pattern cannot be claimed by another pattern, and the current priority of patterns is implemented as follows:

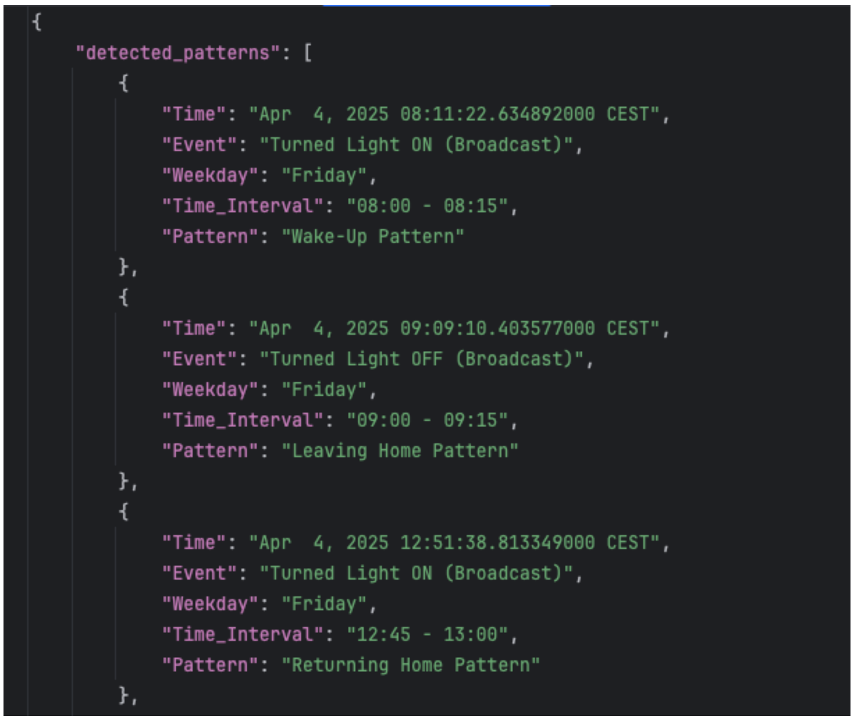
1. Wake-Up Pattern
2. Leaving Home Pattern
3. Returning Home Pattern
4. Midnight Activity Pattern
5. Going to Bed Pattern

This priority implies that a Light ON at e.g., 08:00 in the morning will first be evaluated as a potential Wake-up pattern instead of a Returning Home Pattern, or that a Light OFF which directly follows a Light ON at e.g., 01:00 will first be looked at as a Midnight Activity Pattern instead of a Going to Bed Pattern. If the higher priority claimed the event, it cannot be claimed again by the lower priority. Changing the priorities in the rules_engine.py file would change the order in which events will be claimed by the patterns, which any future user could adjust. In Figure 4.3 an excerpt of an example of a user_profile.json file is shown to demonstrate the pattern detection according to the rules defined in the respective pattern.

Wake-Up Pattern

The Wake-Up Pattern is designed to detect the user's typical morning routine based on early light activity. The detection logic is implemented in the detect_wake_up_patterns() function within the Rules Engine. The pattern aims to identify a reliable signal that the user has woken up based on analyzing Light ON events in the morning. The following conditions must be fulfilled to classify an event as part of Wake-Up Pattern:

1. The Light ON event must occur within a time window between 04:00 and 10:00.
2. Only one wake-up event is recorded per calendar day.
3. At least two Light ON events must occur within 30 minutes for the first Light ON event to be counted as a wake-up event.



```
{
    "detected_patterns": [
        {
            "Time": "Apr  4, 2025 08:11:22.634892000 CEST",
            "Event": "Turned Light ON (Broadcast)",
            "Weekday": "Friday",
            "Time_Interval": "08:00 - 08:15",
            "Pattern": "Wake-Up Pattern"
        },
        {
            "Time": "Apr  4, 2025 09:09:10.403577000 CEST",
            "Event": "Turned Light OFF (Broadcast)",
            "Weekday": "Friday",
            "Time_Interval": "09:00 - 09:15",
            "Pattern": "Leaving Home Pattern"
        },
        {
            "Time": "Apr  4, 2025 12:51:38.813349000 CEST",
            "Event": "Turned Light ON (Broadcast)",
            "Weekday": "Friday",
            "Time_Interval": "12:45 - 13:00",
            "Pattern": "Returning Home Pattern"
        },
```

Figure 4.3: Excerpt of the user_profile.json file with detected patterns taking the merged_events.json file as input.

The selected time window (04:00 - 10:00) is not based on clinical or scientific sources but was chosen to reflect typical wake-up times. These intervals can also be adjusted should different assumptions be preferred in future implementations (e.g., another user of this prototype may consider that waking up only happens after 05:00). The rule that limits detection to a single wake-up event per day is important, as it ensures that subsequent rules in the Rules Engine (e.g., for detecting returning home) can also claim events. While it is possible that a user may briefly wake up and return to sleep before waking again later, this prototype considers only the first qualifying Light ON activity to maintain clarity and avoid overlap. The third condition introduces an additional check to prevent false positives from brief nighttime activity. The assumption is that a genuine wake-up involves multiple light interactions, for example, turning on a bedroom light followed by one in the bathroom or kitchen. Therefore, at least two Light ON events must occur within 30 minutes to validate the detection. Originally, the rule set also included a constraint that no Light OFF event should occur within 15 minutes after the first Light ON. However, this condition was ultimately removed after testing, as it proved too restrictive. Users may turn off their bedroom light shortly after waking up, especially when transitioning to another room.

Leaving Home Pattern

The Leaving Home Pattern aims to detect moments when the user is likely to leave their home, inferred through light-off behavior during defined daytime hours. The detection logic is implemented in the detect_leaving_home_pattern() function and is based on both required time intervals and the absence of follow-up activity. To classify an event as part of a Leaving Home Pattern, the following conditions must be fulfilled:

1. The event must be a Light OFF event.
2. The event must occur within a predefined daytime interval categorized as Morning (04:00 - 11:00), Afternoon (11:00 - 17:00), or Evening (17:00 - 21:00).
3. No further light activity (ON/OFF, saturation, or color control events) should be detected within the 60 minutes following the Light OFF event.

Unlike the Wake-Up Pattern, the Leaving Home Pattern allows for multiple detections per day, as users may leave and return home multiple times. For example, someone may leave in the morning for work, return for lunch, and leave again in the afternoon. The algorithm supports this behavior by scanning all events chronologically and applying the rule without a per-day restriction. The absence of activity following the Light OFF event is a critical indicator in this detection. If no additional events are recorded within one hour, the assumption is that the user has likely left the house. This condition helps differentiate between brief light-off actions within the home and a departure scenario. If any event is detected within 60 minutes, the Light OFF is not considered a departure. The categorization into Morning, Afternoon, and Evening intervals allows later visualization to contextualize departure behavior across different times of day. This design is intended to improve interpretability in the final dashboard and make recurring behavior patterns more evident. The rule doesn't consider the specific room or light source, as the prototype lacks device-specific commands. But it is important to mention that this simplification may lead to edge cases (e.g., a user turning off a hallway light while remaining at home but not interacting with the light system anymore for over one hour).

Returning Home Pattern

The Returning Home Pattern is designed to detect when the user arrives back home, based on light activity in the form of Light ON commands, color control, or saturation control events. The logic for detecting this pattern is implemented in the detect_returning_home_pattern() function. This rule assumes that switching on lights or adjusting their properties in specific time intervals, after a period of inactivity, signals that the user has entered their home. The following conditions are applied to determine such an event:

1. The event must be either a Light ON, Saturation Control, or Color Control event.
2. The event must occur within a predefined time interval assigned to either Morning (07:00 - 11:00), Afternoon (11:00 - 17:00), or Evening (17:00 - 23:59).
3. No other events (of any type) must have occurred within the previous 60 minutes.

This rule, like the Leaving Home Pattern, allows multiple detections per calendar day, allowing the user to leave and return to their home several times (e.g., returning for lunch or after work in the evening). The requirement for a 60-minute quiet period prior to the ON or control events serves to reduce false positives, such as lights being turned on from another room as part of normal movement in the home.

Going to Bed Pattern

The Going to Bed Pattern is intended to detect the user's nightly routine, particularly the moment they turn off their lights before going to sleep. This pattern is implemented in the detect_going_to_bed_pattern() function. It analyzes Light OFF events in their temporal context to determine whether they indicate the time the user goes to sleep. To qualify as a Going to Bed event, the following conditions must be satisfied:

1. The event must be a Light OFF command.
2. It must occur within the defined bedtime interval 21:00 to 03:00.
3. There must be no other recorded events in the 60 minutes following the Light OFF event.
4. Only one Going to Bed event is stored per calendar date. If multiple valid events exist, only the last one of the day is recorded.

The 60-minute inactivity condition ensures that the detected event is not followed by additional interaction. This condition is crucial for filtering out false positives and increasing the likelihood that the event genuinely marks the transition to bedtime. Only the last eligible Light OFF event per day is stored, assuming it represents the final user interaction with the lightning system before sleep. Like in the detection for the Wake-Up Pattern, the selected bedtime window between 21:00 and 03:00 is not derived from clinical or scientific sources but was instead chosen to reflect a broad range of common bedtimes. This range tries to include both early and late-night routines. But this parameter can be easily adapted in future implementations to reflect different assumptions or cultural routines.

Midnight Activity Pattern

The Midnight Activity Pattern detects brief periods of light usage that occur during typical sleeping hours, which may indicate short interruptions in the user's sleep, such as using the bathroom or getting a glass of water. The detection logic is implemented in the detect_midnight_activity_pattern() function. Events or sequences are classified as a Midnight Activity Pattern if they fulfill the following conditions:

1. A Light ON, Saturation Control, or Color Control event occurs between 00:00 and 04:00.

2. This event is followed by the Light OFF event within the next 15 minutes.
3. Multiple detections per day are allowed (i.e., each brief activity during the night is counted independently).

These conditions help filter out brief nighttime light usage while avoiding misclassification of longer nighttime routines or patterns that may instead indicate a bedtime routine. The selected time window between 00:00 and 04:00 reflects commonly assumed hours of sleep and was chosen based on practical assumptions rather than scientific studies. Future users may adjust these intervals to tailor the detection logic to different lifestyles or sleep norms.

4.3.2 Patterns Probability Script

Once the behavioral patterns are detected and recorded in the user_profile.json file, a dedicated script, pattern_probabilities.py, processes this data to provide a statistical overview of recurring behavior. The goal is to analyze how consistently a user conducts a specific behavior (e.g., waking up at a similar time every Tuesday) and to visualize these regularities over time.

Introduction of Dynamic Time Interval Detection

As discussed in the design section, all detected events were grouped into fixed 15-minute intervals (e.g., 07:00-07:15 or 08:30-08:45) based on predefined time intervals. While this structure allows for standardization and comparison between events, it introduces limitations. Especially if a user, for example, wakes up repeatedly between 07:24 and 07:36, these two events would be split into the time intervals 07:15-07:30 and 07:30-07:45, even though less than 15 minutes separate them. To address this, it was discussed to extend the system with a dynamic time interval detection. Instead of relying on fixed 15-minute intervals, a flag 'USE_DYNAMIC_INTERVALS' was introduced that can be set to either *True* or *False* before running the pattern_probabilities.py script. If the flag is set to *False*, the fixed 15-minute time intervals are used. If the flag is set to *True*, the implementation extracts the exact time of each event and groups similar times based on proximity. The current threshold for grouping is set to 15 minutes, meaning events that occur within a 15-minute window are treated as part of the same group. This threshold can be adjusted by the user of the prototype, as seen in Figure 4.4 in the pattern_probabilities.py script.

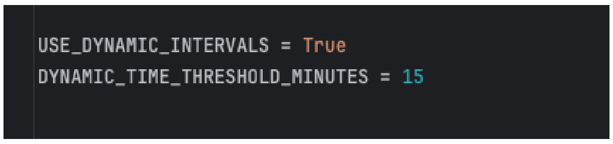
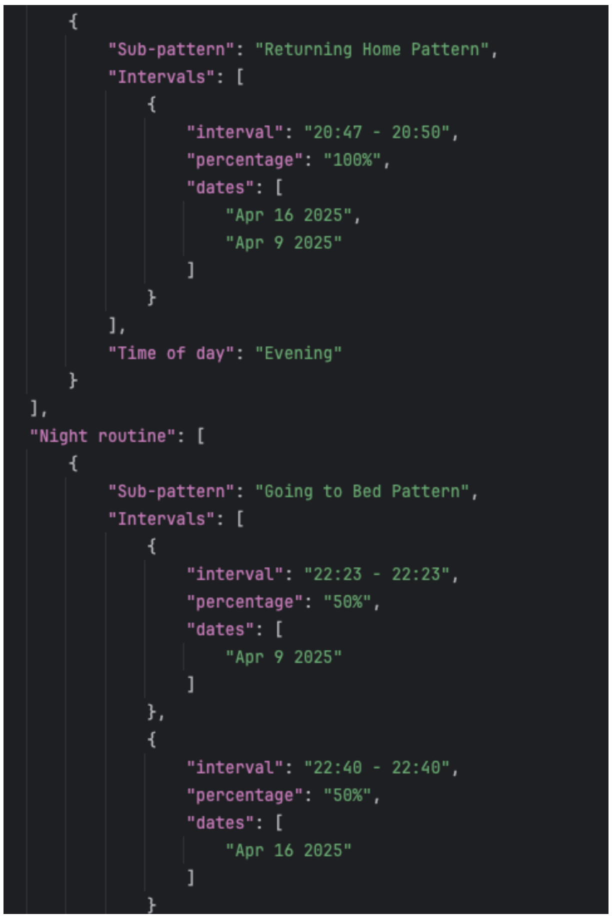

```
USE_DYNAMIC_INTERVALS = True
DYNAMIC_TIME_THRESHOLD_MINUTES = 15
```

Figure 4.4: Dynamic Time Threshold

This approach allows for a more natural and data-driven aggregation and gives a clearer view of how tightly the user's routines are clustered. For example, three wake-up events at

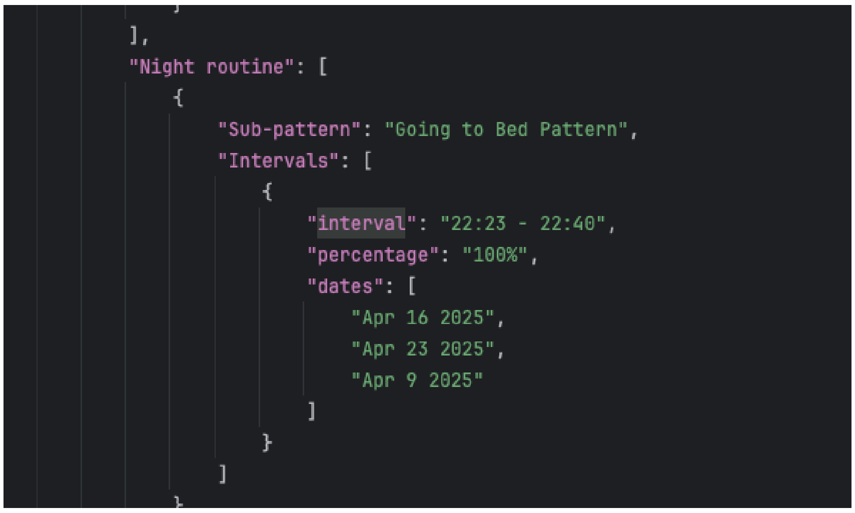
07:25, 07:28, and 07:33 would be dynamically grouped and reported as one range, 07:25-07:33, with a percentage indicating the consistency of that behavior. An example with the 15-minute threshold applied is shown in Figure 4.5 for the weekday Wednesday. A dynamic interval 20:47 - 20:50 was retrieved for the Returning Home Pattern and applied a percentage of 100%. But the Going to Bed Pattern is split into two time intervals, as the events contain more than a 15-minute time difference and are assigned a probability of 50%.



```
    {
        "Sub-pattern": "Returning Home Pattern",
        "Intervals": [
            {
                "interval": "20:47 - 20:50",
                "percentage": "100%",
                "dates": [
                    "Apr 16 2025",
                    "Apr 9 2025"
                ]
            }
        ],
        "Time of day": "Evening"
    }
],
"Night routine": [
    {
        "Sub-pattern": "Going to Bed Pattern",
        "Intervals": [
            {
                "interval": "22:23 - 22:23",
                "percentage": "50%",
                "dates": [
                    "Apr 9 2025"
                ]
            },
            {
                "interval": "22:40 - 22:40",
                "percentage": "50%",
                "dates": [
                    "Apr 16 2025"
                ]
            }
```

Figure 4.5: Weekday Wednesday example with 15-minute threshold

To further exemplify the dynamic time interval approach, a third Wednesday Going to Bed Time was collected at 22:32 which lies between the already two collected times at 22:23 and 22:40. The algorithm groups these three times now into one time interval, as less than 15 minutes separate 22:23 from 22:32 and 22:32 from 22:40 as seen in Figure 4.6. The interval is now assigned a percentage of 100%.



```
            ],
            "Night routine": [
                {
                    "Sub-pattern": "Going to Bed Pattern",
                    "Intervals": [
                        {
                            "interval": "22:23 - 22:40",
                            "percentage": "100%",
                            "dates": [
                                "Apr 16 2025",
                                "Apr 23 2025",
                                "Apr 9 2025"
                            ]
                        }
                    ]
```

Figure 4.6: Adjusted time interval for Wednesday with third time collected.

Implementation Overview

The script reads all detected patterns from the user_profile.json file and processes them by weekday and pattern type. For each weekday (e.g., ”Monday”) and each behavior (e.g., ”Wake-Up Pattern”), the system calculates how often events occur within specific time ranges. The flow of the script is as follows:

1. Read and sort all detected events per weekday and behavior type.
2. Group timestamps into time intervals, using the two methods:

   Static Mode: Using fixed 15-minute intervals (e.g., 08:00-08:15).

   Dynamic Mode: Extracts exact HH:MM timestamps and groups them using the proximity-based threshold (default is 15 minutes).
3. Count occurrences of each grouped interval, and calculate their relative frequency as a percentage over the total number of days that the weekday was observed.
4. Output the result as a structured pattern_probabilities.json file, which includes:
   (a) All observed intervals
   (b) Percentages indicating behavioral consistency
   (c) Dates on which the pattern occurred
   (d) Metadata: Observed date range and the mode used (static or dynamic)

4.3.3 Choice of Visualization Framework

To present the inferred behavior patterns in a structured way, a web-based dashboard was implemented. The goal was to present a clear and interactive visualization of recurring patterns. Rather than building a desktop application, a browser-based interface was chosen to ensure the ease of development, maintainability, and platform independence.

Among the available Python libraries for creating user interfaces and dashboards, Streamlit was chosen for the following reasons:

- Integration with Python: Since the prototype is implemented in Python, Streamlit allowed a seamless integration without having to switch to another language.
- Built-in Tools for Data Visualizations: Streamlit allows the usage of visualization libraries like plotly, which was also used at a later point.
- Web-Based Deployment: The application runs locally in a web browser without having to install any separate GUI components or software.
- Support and Documentation: The Streamlit documentation is well-organized and well-explained, with tutorials available to provide aid for future users and developers of this prototype.

Pattern Probability Dashboard

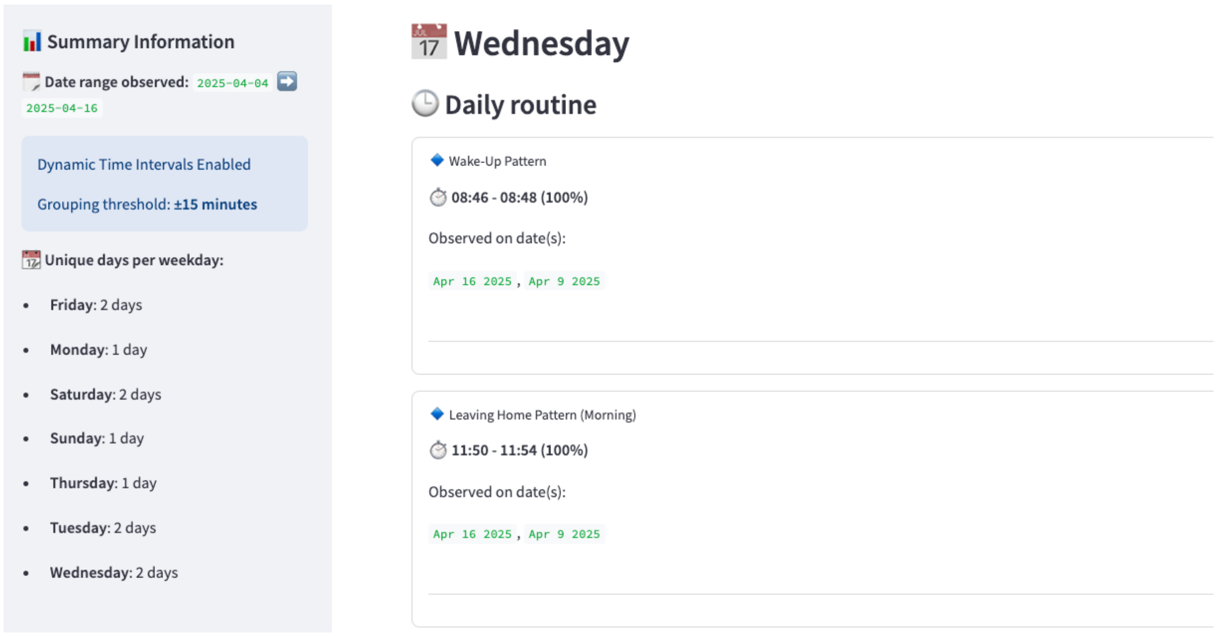
To visualize the recurring behavior patterns detected by the user profiling system, a Streamlit-based dashboard was implemented using the pattern_dashboard.py script. This dashboard provides an intuitive interface for exploring the output of the system, the pattern_probabilities.json. The dashboard loads the results from the JSON file and displays them in an interactive format. For each weekday, the interface presents the top routines, Daily Routine and Night Routine, along with their corresponding sub-patterns (e.g., Wake-Up Pattern, Returning Home Pattern etc). For every detected sub-pattern, the system shows:

- The time interval(s) in which the behavior typically occurred.
- The frequency, displayed as a percentage of how often this pattern occurred on that weekday.
- The specific dates when the behavior was detected.

The following additional information is provided in a sidebar:

- The date range over which observations were made.
- The number of unique days per weekday used for the analysis.
- Whether dynamic time interval grouping was enabled, and if so, the exact threshold in minutes used for grouping similar times.

Additionally, an expandable section explains the logic behind each behavioral pattern, providing transparency on how the system interprets user routines based on the detected Philips Hue commands. An excerpt of the dashboard is shown in Figure 4.7.


Summary Information

Date range observed: 2025-04-04 ➡ 2025-04-16

Dynamic Time Intervals Enabled

Grouping threshold: ±15 minutes

Unique days per weekday:

- **Friday:** 2 days
- **Monday:** 1 day
- **Saturday:** 2 days
- **Sunday:** 1 day
- **Thursday:** 1 day
- **Tuesday:** 2 days
- **Wednesday:** 2 days

Wednesday

Daily routine

Wake-Up Pattern

08:46 - 08:48 (100%)

Observed on date(s):

Apr 16 2025 , Apr 9 2025

Leaving Home Pattern (Morning)

11:50 - 11:54 (100%)

Observed on date(s):

Apr 16 2025 , Apr 9 2025

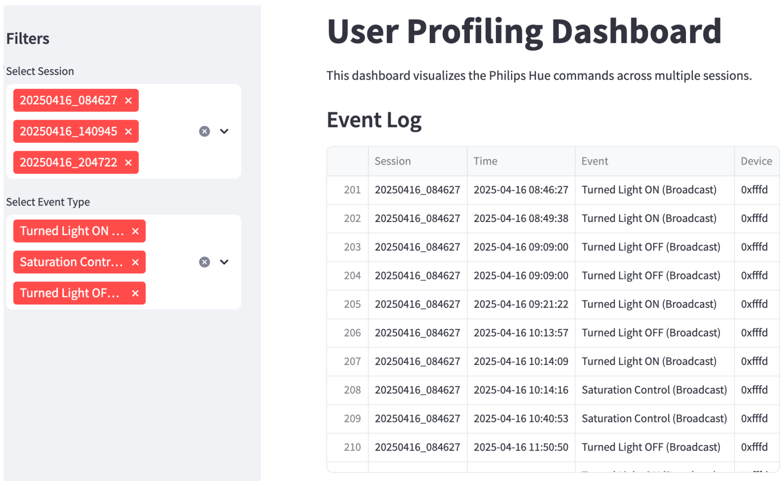
Figure 4.7: Excerpt of Dashboard in Streamlit

4.3.4 Additional Session-Level Visualization Dashboard

In addition to the pattern-level aggregation presented in the Pattern Probability Dashboard, another Streamlit-based dashboard was developed to allow further insights and analysis of the individual sessions, the events_dashboard.py. This component takes the merged_events.json file as input, which is introduced in the section 'CSV to JSON Script'. The dashboard provides an interactive graphical user interface that enables the filtering and comparison of individual sessions. The features include:

- **Session Selection:** With a dynamic sidebar, the user can select multiple sessions to focus on for their analysis.
- **Event Type Filtering:** The user can select which events (e.g., Light ON, Light OFF, Saturation Control) are displayed, depending on which events they want to focus on.
- **Tabular Event Log View:** A scrollable table displays all filtered events chronologically, including timestamps and associated end device information.
- **Event Count Summaries:** Two bar charts summarize the total number of events per session and the frequency of each event type across the selected data.
- **Temporal View of Events:** An event timeline displays the sequence of events, providing a visual representation of when they occurred.

The dashboards of the events_dashboard.py are displayed below with the three selected sessions 20250416_084627, 20250416_140945, and 20250416_204722 from April 16, 2025.


Filters

Select Session

20250416_084627 ×
20250416_140945 ×
20250416_204722 ×

Select Event Type

Turned Light ON … ×
Saturation Contr… ×
Turned Light OF… ×

User Profiling Dashboard

This dashboard visualizes the Philips Hue commands across multiple sessions.

Event Log

	Session	Time	Event	Device
201	20250416_084627	2025-04-16 08:46:27	Turned Light ON (Broadcast)	0xfffd
202	20250416_084627	2025-04-16 08:49:38	Turned Light ON (Broadcast)	0xfffd
203	20250416_084627	2025-04-16 09:09:00	Turned Light OFF (Broadcast)	0xfffd
204	20250416_084627	2025-04-16 09:09:00	Turned Light OFF (Broadcast)	0xfffd
205	20250416_084627	2025-04-16 09:21:22	Turned Light ON (Broadcast)	0xfffd
206	20250416_084627	2025-04-16 10:13:57	Turned Light OFF (Broadcast)	0xfffd
207	20250416_084627	2025-04-16 10:14:09	Turned Light ON (Broadcast)	0xfffd
208	20250416_084627	2025-04-16 10:14:16	Saturation Control (Broadcast)	0xfffd
209	20250416_084627	2025-04-16 10:40:53	Saturation Control (Broadcast)	0xfffd
210	20250416_084627	2025-04-16 11:50:50	Turned Light OFF (Broadcast)	0xfffd

Figure 4.8: Session and Event Type Selection and Display of Event Logs Table

Figure 4.8 showcases the session and event type selection on the sidebar. All events are displayed in a scrollable list **Event Log**, listing the session, date and time, the event, and the end device.

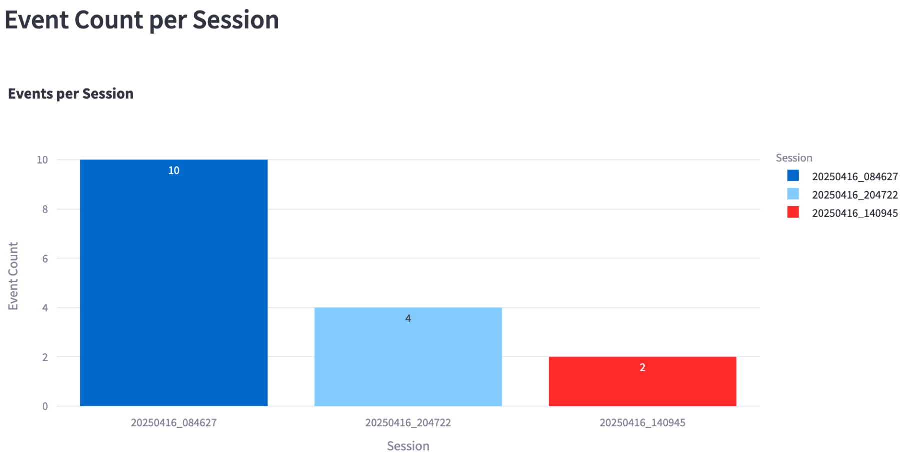


Figure 4.9: Event Count per Session

Figure 4.9 demonstrates the event count by session. Therefore, the number of commands made by the user can be easily derived.

In Figure 4.10, the event frequency is displayed. It can therefore be seen how many Light ON commands, how many Light OFF commands, or how many Saturation Control commands were made by the user during the selected sessions.

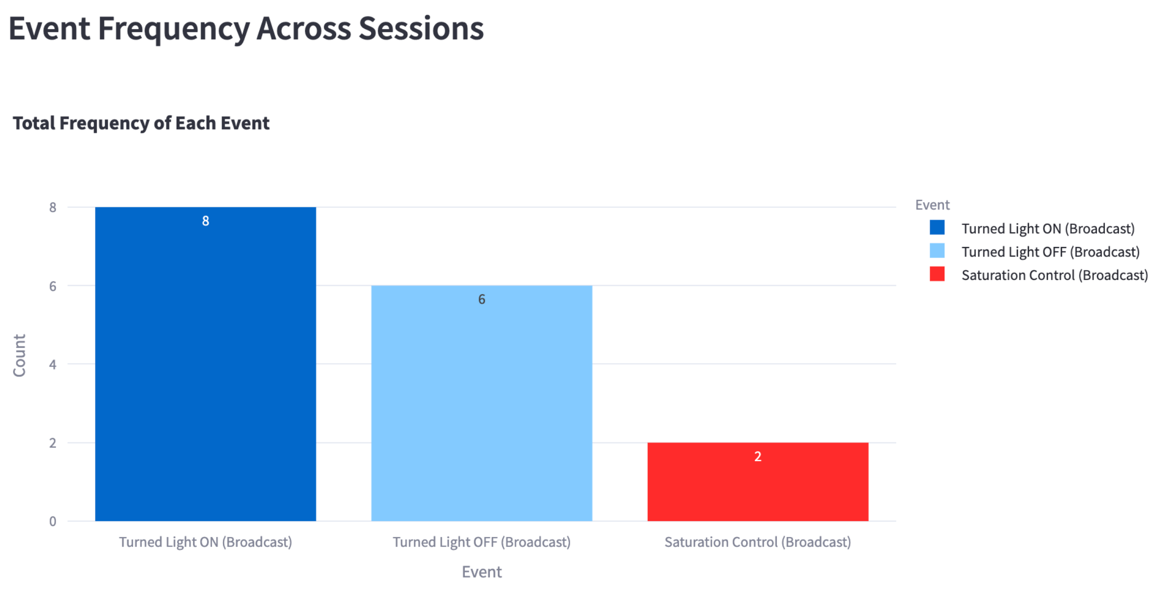


Figure 4.10: Event Frequency Across Sessions

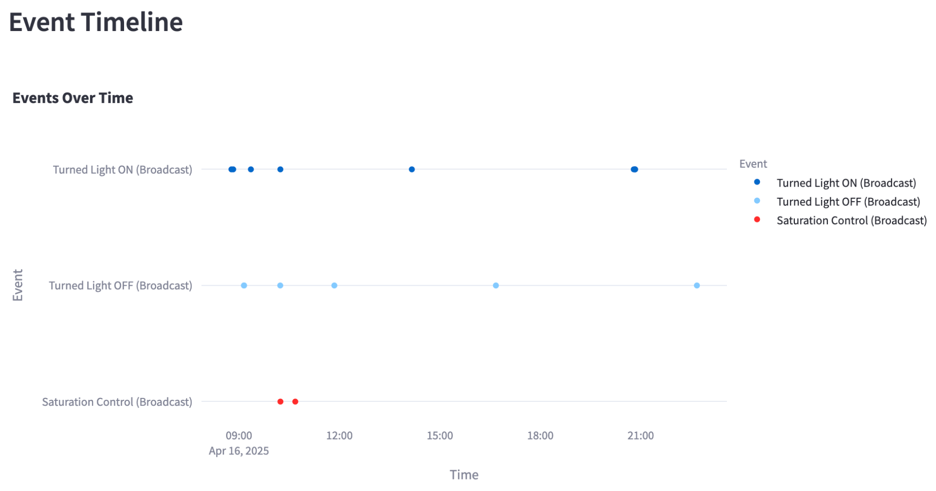


Figure 4.11: Event Timeline

The event timeline is depicted in Figure 4.11. With the time being displayed on the x-axis, it can be visualized when the user made the commands. By hovering over the points, the exact time is displayed.

By running the events_dashboard.py script in Streamlit, all sessions will be pre-selected at first, providing an overview of all sessions over time. In Figure 4.12 an overview of all events per sessions is shown.

In Figure 4.13, the total frequency of the captured events across all captured sessions can be seen. From Figure 4.13, it can be derived that the Turned Light OFF command was used most by the user, followed by the Turned Light ON command. The command Saturation Control is the least used command.

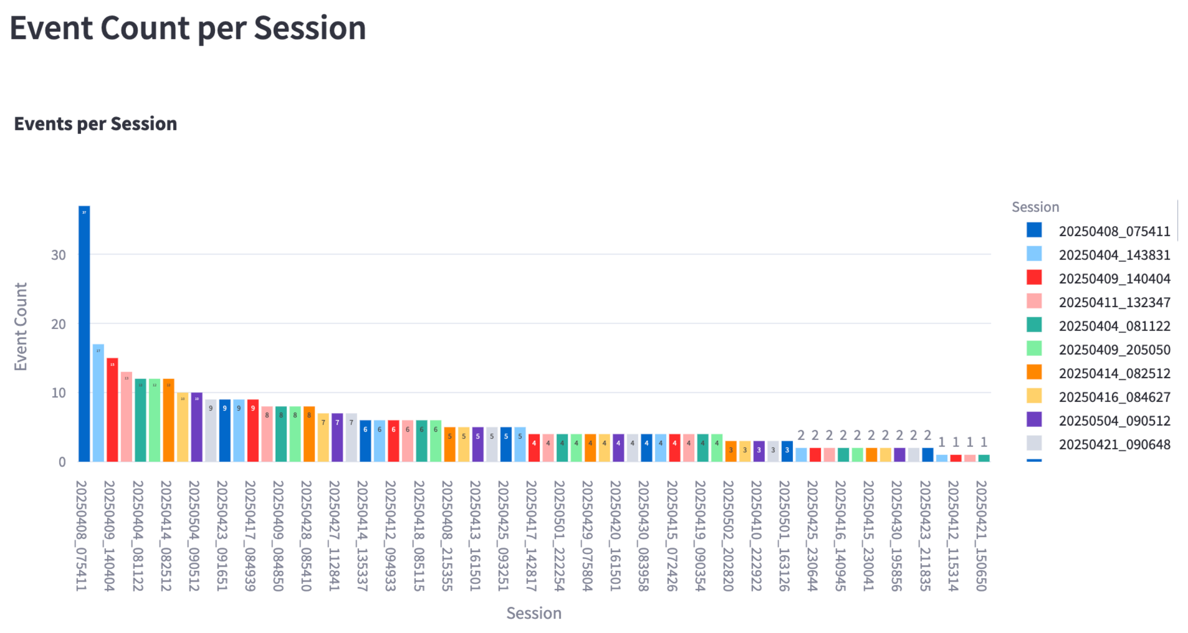


Figure 4.12: Events per session of all captured sessions

Also the event timeline may be used with all captured sessions selected as seen in Figure 4.14. The x-axis now displays dates rather than the time as seen in Figure 4.11. While the Light ON and OFF command were used very regularly, it can be seen with this dashboard that for example the Saturation Control command was not used at all between April 20 and April 27.

4.3.5 Outliers Detection Script

While the previously described pattern detection script and the session-level visualization dashboard provide insight into recurring behaviors, their probabilities, and details of individual sessions, it was discussed to implement a detection system to identify behavioral shifts and anomalies. Therefore, an additional script outliers_detection.py was implemented to identify events where the user conducts unusual behavior on a specific day compared to their typical routine. For example, if a user usually wakes up at around 7:00 on Mondays but wakes up at 9:30 on a particular Monday. This deviation may indicate a change in routine due to other factors, such as illness or vacation. The script takes the user_profile.json as input and outputs a behavioral_shift.json. The script currently analyzes the patterns "Wake-Up" and "Going to Bed" since they both contain only one event per day. More patterns can be added to the parameter PATTERNS_TO_ANALYZE, but the script would require a slight adjustment, as the Returning and Leaving Home patterns contain several events per day, necessitating accurate calculation and presentation of the results in a meaningful way. Due to time constraints, these adjustments were left out. The detection flow is as follows:

1. **Selection of patterns to analyze:** At the moment the script only analyzes the Wake-Up pattern and the Going to Bed pattern.

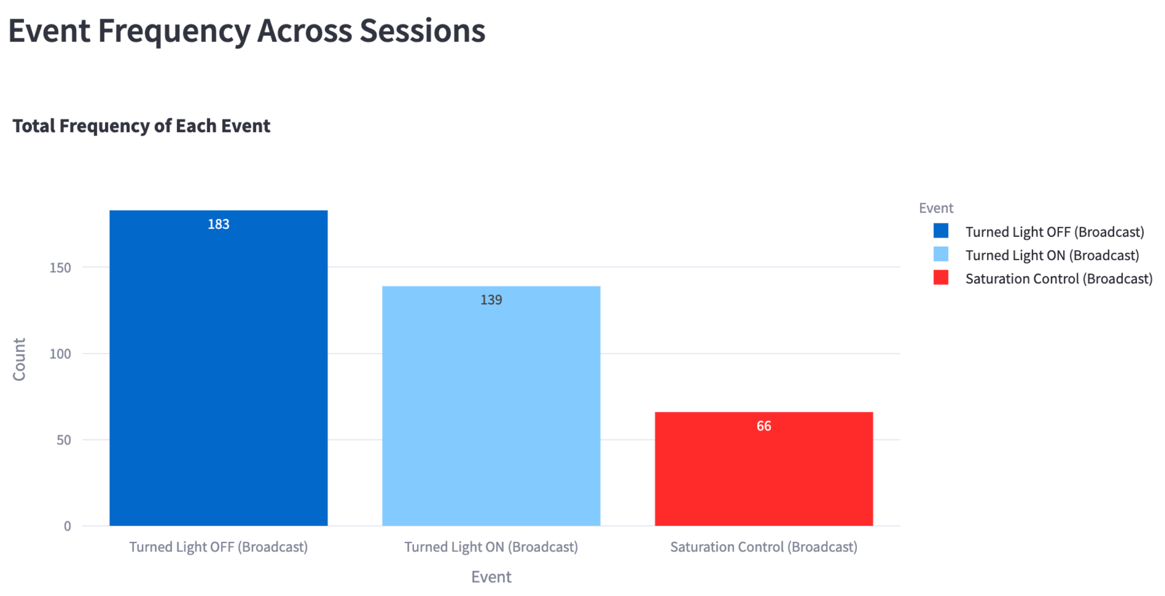


Figure 4.13: Total frequency of each event across all captured sessions

2. **Data Grouping:** For each pattern, the system groups the detected events by weekday and collects the event times.
3. **Median Calculation:** The typical routine time for each pattern on each weekday is defined using the median time.
4. **Deviation Threshold:** The threshold is currently set to 60 minutes but can be adjusted with the parameter DEVIATION_THRESHOLD_MINUTES. If the observed time on a particular day differs from the calculated median by more than the deviation threshold a deviation is marked.
5. **Minimum Data Requirement:** With the parameter MINIMUM_REQUIRED_DAYS, the required number of days can be set to ensure reliability. For testing purposes of this thesis, the minimum required days are set to 3.

Outliers Detection Dashboard

An additional script outliers_dashboard.py was implemented to visualize the generated file behavioral_shift.json by the previously explained outliers_detection.py. An excerpt of the dashboard is shown in Figure 4.15.

4.3.6 Long-term Habit Trend Analysis Script

In addition to identifying recurring behavior patterns, an additional script was implemented to analyze long-term trends in the user's routine. The purpose and goal were to detect significant shifts in the timing of recurring habits. For example, a user usually wakes up at 07:00, but their wake-up time is shifted later after a few weeks. Such

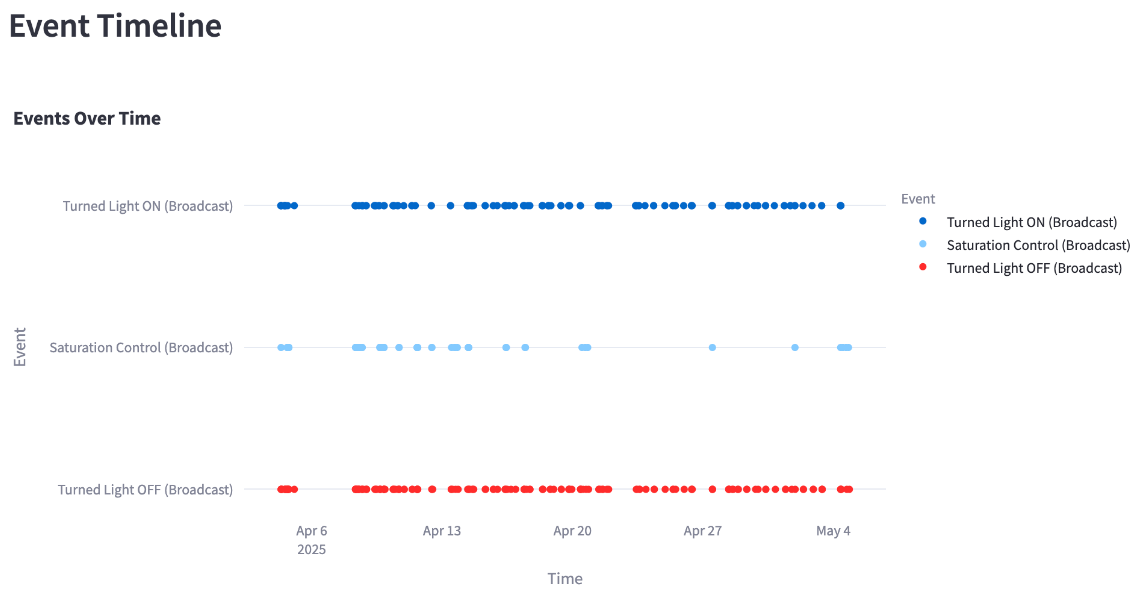


Figure 4.14: Event timeline with all captured sessions selected

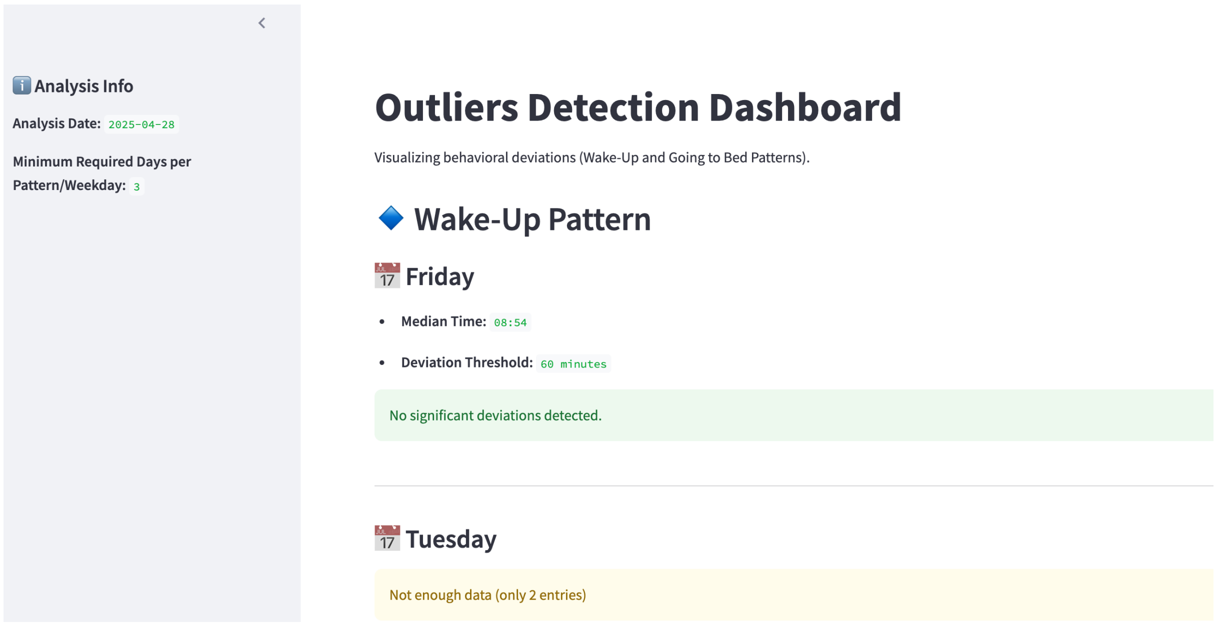
deviations could indicate lifestyle changes or adaptations in the user's daily routine over time. Therefore, the long-term analysis focuses on identifying gradual behavioral shifts. Unlike the immediate detection of outliers (as identified by the outliers_detection.py), this approach analyzes the data over time to uncover stable trends rather than single anomalies.

The script long_term_trends.py takes the user_profile.json file as input. It aggregates and analyzes the time-based data grouped by Weekday, behavioral pattern, and ISO Calendar Week (e.g., 2025-W14). For each week, the system calculates the average event time (in minutes since midnight) for each pattern and weekday combination. The process includes the following steps:

1. **Grouping Event Times by Week** and extracting and converting the event time into minutes since midnight for later analysis.

2. **Compute Weekly Averages:** For each week where a given pattern occurs, the average time of these events is calculated.

3. **First and Second Half Comparison:** After collecting the weekly averages, the collected time series is split into two halves. The first half represents the first half of the observation period. In conclusion, the second half the later weeks of the observation period. The mean average time is calculated separately for both halves. A trend is then determined based on the difference between the two averages. If the difference is less than 30 minutes, the habit is considered ”stable”. If the second half is later by more than 30 minutes than the first half, the habit is classified as ”shifted later”. Vice versa, if the second half is earlier by more than 30 minutes than the first half, the habit is classified as ”shifted earlier” as seen in Listing 4.2. The 30-minute threshold can be adapted by any future user who considers another threshold more applicable for this analysis.


ℹ️ Analysis Info

Analysis Date: 2025-04-28

Minimum Required Days per Pattern/Weekday: 3

Outliers Detection Dashboard

Visualizing behavioral deviations (Wake-Up and Going to Bed Patterns).

🔷 Wake-Up Pattern

📅 Friday

- Median Time: 08:54
- Deviation Threshold: 60 minutes

No significant deviations detected.

📅 Tuesday

Not enough data (only 2 entries)

Figure 4.15: Outliers Detection Dashboard

```
mid_point = len(weekly_averages) // 2
    first_half_avg = statistics.mean(weekly_averages[:mid_point
       ])
    second_half_avg = statistics.mean(weekly_averages[mid_point
       :])

    if abs(second_half_avg - first_half_avg) < 30: # Less than
       30min = stable
        trend = "stable"
    elif second_half_avg > first_half_avg:
        trend = "shifted later"
    else:
        trend = "shifted earlier"
```

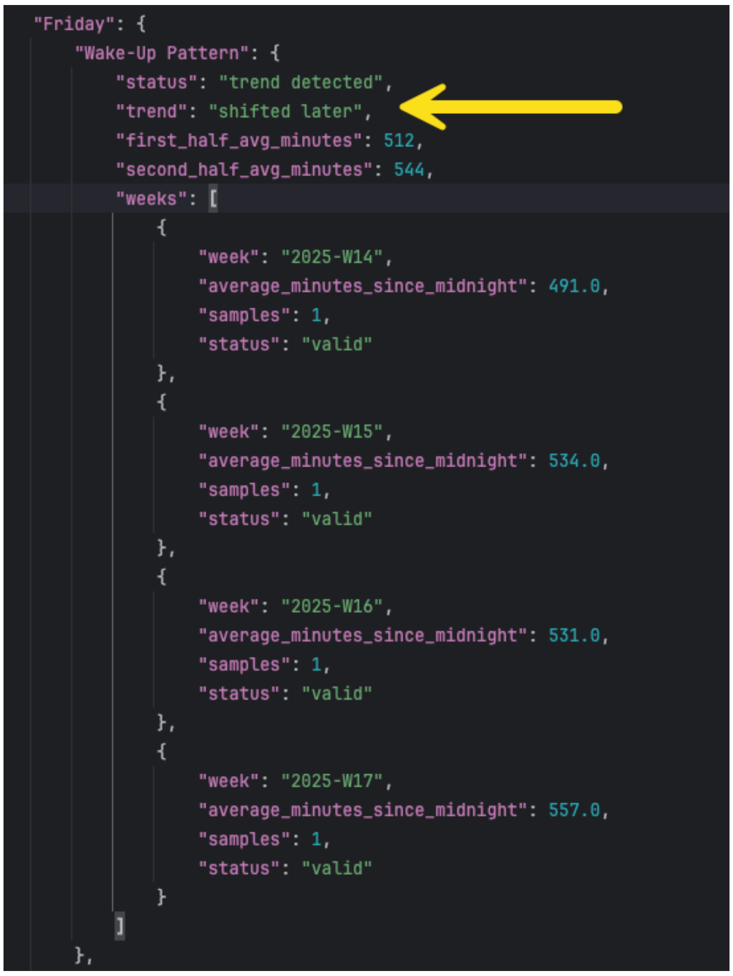
Listing 4.2: First and Second Half Comparison

4. **Minimum Data Requirement:** The parameter MIN_WEEKS_FOR_TREND_ANALYSIS marks how many weeks are required for a meaningful trend detection and can be easily adjusted by the user. For testing purposes of this thesis, it is set to 3.

The script outputs a file called habit_trends.json, of which an excerpt is shown in Figure 4.16.

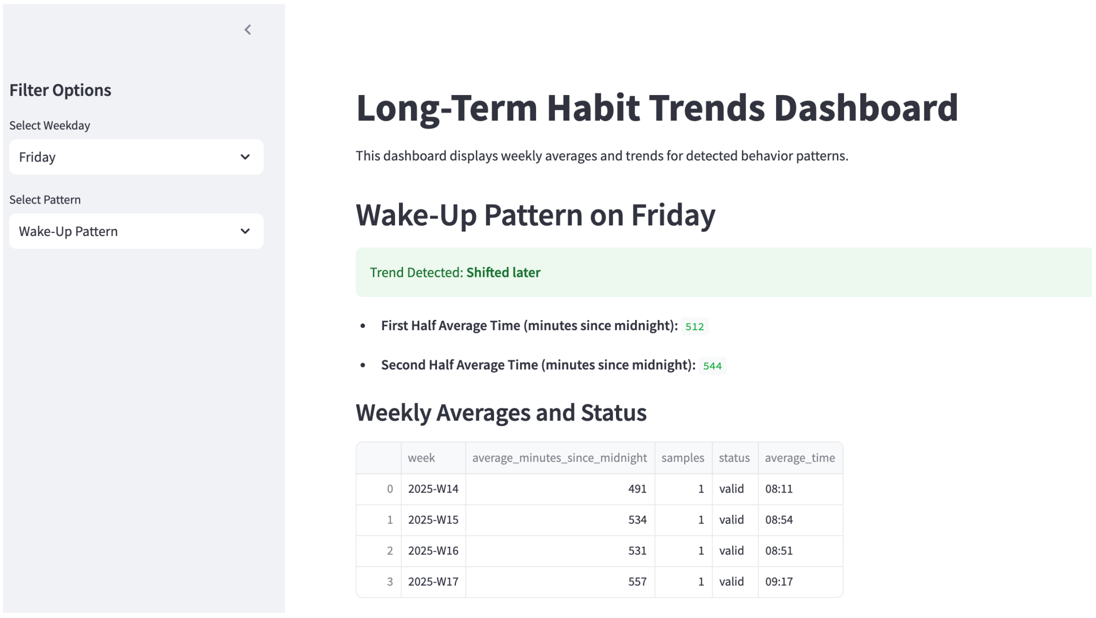
Long-Term Habit Trends Dashboard

To visualize the habit_trends.json file, another Streamlit dashboard was implemented to portray the long-term habit tracking trends. The day of the week, as well as the pattern, can be selected to provide a better overview. A summary of the detected trend is portrayed in Figure 4.17.



```
"Friday": {
    "Wake-Up Pattern": {
        "status": "trend detected",
        "trend": "shifted later",
        "first_half_avg_minutes": 512,
        "second_half_avg_minutes": 544,
        "weeks": [
            {
                "week": "2025-W14",
                "average_minutes_since_midnight": 491.0,
                "samples": 1,
                "status": "valid"
            },
            {
                "week": "2025-W15",
                "average_minutes_since_midnight": 534.0,
                "samples": 1,
                "status": "valid"
            },
            {
                "week": "2025-W16",
                "average_minutes_since_midnight": 531.0,
                "samples": 1,
                "status": "valid"
            },
            {
                "week": "2025-W17",
                "average_minutes_since_midnight": 557.0,
                "samples": 1,
                "status": "valid"
            }
        ]
    },
```

Figure 4.16: Trend ”shifted later” detected for Friday Wake-Up Pattern



Filter Options

Select Weekday: Friday

Select Pattern: Wake-Up Pattern

Long-Term Habit Trends Dashboard

This dashboard displays weekly averages and trends for detected behavior patterns.

Wake-Up Pattern on Friday

Trend Detected: Shifted later

- First Half Average Time (minutes since midnight): 512
- Second Half Average Time (minutes since midnight): 544

Weekly Averages and Status

	week	average_minutes_since_midnight	samples	status	average_time
0	2025-W14	491	1	valid	08:11
1	2025-W15	534	1	valid	08:54
2	2025-W16	531	1	valid	08:51
3	2025-W17	557	1	valid	09:17

Figure 4.17: Excerpt of the Long-Term Habit Trends Dashboard (Summary) for Wake-Up Pattern on Friday

Additionally, the trend is also visualized in a timeline as seen in Figure 4.18.

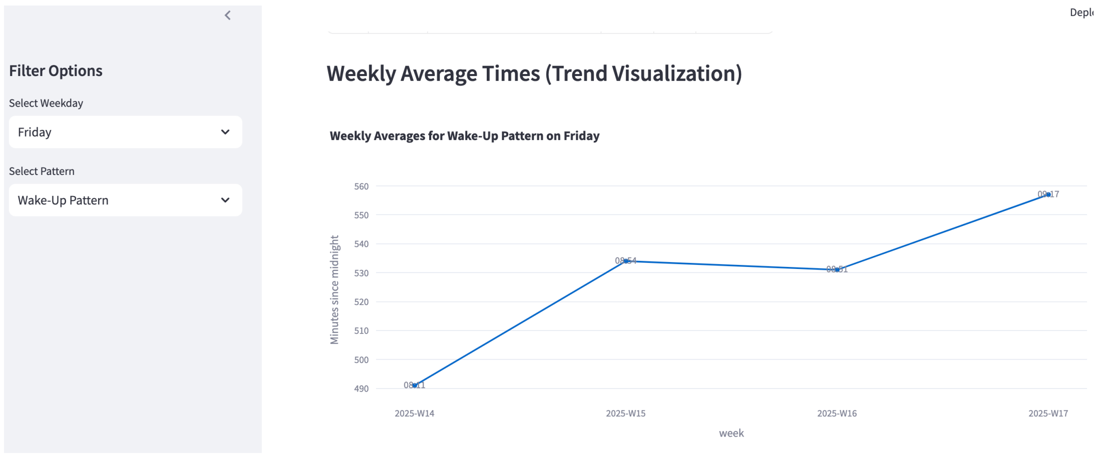


Figure 4.18: Excerpt of the Long-Term Habit Trends Dashboard (Timeline) for Wake-Up Pattern on Friday

Chapter 5

Evaluation

In the following section, the developed user profiling system is evaluated with a focus on the accuracy of the detected behavior patterns, recurring routines, and the ability to track long-term habit changes.

5.1 Test Scenarios

The evaluation is based on Philips Hue commands collected over approximately four weeks, from April 4, 2025, to May 4, 2025, across multiple sessions.

Experiments with different durations were conducted to evaluate the system's effectiveness:

1. Test each pattern once: Ensuring that each pattern is at least tested once and evaluated for its correctness.
2. Test one full weekday: Test at least one full weekday to ensure the patterns work correctly with each other and don't overlap.
3. Test one week (work week + weekends): Test at least one week to see differences between different weekdays and the weekend.

To evaluate whether probabilities are calculated correctly in the pattern_probabilities.py script, at least two days of the same weekday were necessary, as probabilities otherwise always stay at 100%. It was therefore decided to collect data for as long as possible to evaluate the correctness of the pattern_probabilities.py script. To detect at least one outlier with outliers_detection.py script, at least three days of the same weekday were necessary. The development of the long_term_trends.py script further amplified the necessity of a large data set, which is why, finally, data capture of approximately four weeks was evaluated.

5.2 Data Capture

It is important to note that the data collected from April 4, 2025, to May 4, 2025, was not captured in a single session. Individual days are, at times, also divided into multiple sessions due to the author's schedule at home. Wireshark would also stop recording a session if the screen of the device on which Wireshark runs locks after inactivity. Therefore, constant input is required to keep the session active. Long sniffing sessions also proved heavy on the test device and required high CPU usage when analyzing the sniffing sessions with the prototype built by [6]. The sniffing sessions were therefore captured whenever the author of this thesis was at home during this period.

5.3 Evaluation of Rules Engine and Sub-Patterns

Capturing data started on 04.04.2025 to see if the pattern detection works correctly and each pattern gets detected. The 04.04.2025 is segmented into four different sessions:

- session_id "20250404_081122": Containing 12 commands.
- session_id "20250404_125138": Containing 7 commands.
- session_id "20250404_143831": Containing 14 commands.
- session_id "20250404_182832": Containing 7 commands.

An overview of all sessions can be found in A.1 and A.2, but to provide a short overview to picture the sessions better, the timeline of April 4, 2025, is depicted in Figure 5.1 created with the events_dashboard.py script in Streamlit.

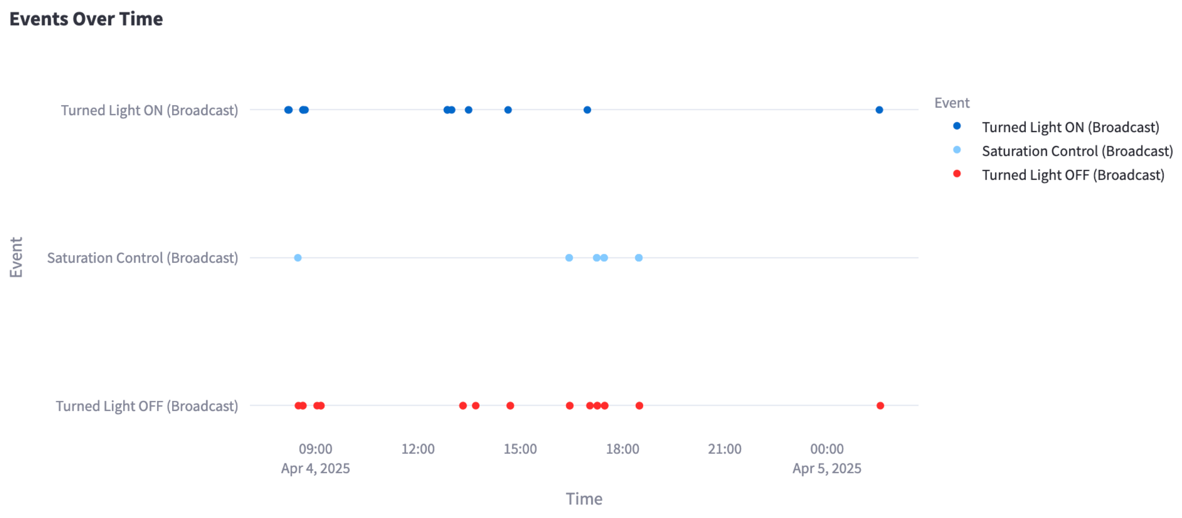


Figure 5.1: Timeline of April 4, 2025, across the aforementioned four sessions

After running the rules_engine.py script, it correctly identified the following patterns:

Time	**Pattern**	**Event Type**	**Reason for Detection**
08:11:22	Wake-Up	Light ON	First Light ON event of the day, followed by another ON at 08:12:59
09:09:10	Leaving Home	Light OFF	No further activity detected for at least 60 minutes after this OFF
12:51:38	Returning Home	Light ON	No activity was recorded for 60+ minutes before this ON
14:42:25	Leaving Home	Light OFF	No further activity for 60+ minutes after this event
16:25:58	Returning Home	Saturation Control	No activity for 60+ minutes prior to this event
17:28:26	Leaving Home	Light OFF	No activity detected afterward for over 60 minutes
18:28:32	Returning Home	Saturation Control	No activity for 60+ minutes prior to this event
18:29:32	Leaving Home	Light OFF	No activity for 60+ minutes following this OFF
01:31:45 (April 5)	Midnight Activity	Light ON	Light ON between 00:00 and 04:00 followed by OFF within 15 minutes
01:33:31 (April 5)	Midnight Activity	Light OFF	OFF within 15 minutes of previous ON event

Table 5.1: Detected Patterns and Conditions on April 4, 2025

The Going to Bed Pattern was tested for the first time on April 8, 2025. To not repeat and re-explain the detection of the other patterns, only the last session with session id ”20250408_215355”, which contains the Going-to-Bed Pattern is investigated in the following Table 5.2:

Time	**Pattern**	**Event Type**	**Reason for Detection**
21:53:55	Returning Home	Light ON	No activity for 60+ minutes prior to this event
22:40:45	Going to Bed	Light OFF	The last Light OFF detected between 21:00 and 03:00, followed by no further activity

Table 5.2: Detected Patterns and Conditions on April 8, 2025, in session ”20250408_215355”

5.4 Evaluation of Pattern Probabilities

In order to evaluate the accuracy of the pattern probability analysis (pattern_probabilities.py), the weekday Wednesday was selected as it contains four full data points for all the patterns, except the Midnight Activity Pattern. Table 5.3 summarizes the recurring patterns detected for this day, based on data collected between April 9 and April 30, 2025.

Sub-pattern	**Time of Day**	**Interval**	**Frequency**	**Observed Dates**
Wake-Up Pattern	Morning	08:39 –08:48	75%	Apr 9, Apr 16, Apr 30
		09:16 –09:16	25%	Apr 23
Leaving Home Pattern	Morning	08:57 –08:57	25%	Apr 30
		11:11 –11:11	25%	Apr 23
		11:50 –11:54	50%	Apr 9, Apr 16
Leaving Home Pattern	Afternoon	14:20 –14:20	25%	Apr 23
		16:40 –16:40	50%	Apr 9, Apr 16
Returning Home Pattern	Afternoon	13:20 –13:20	25%	Apr 23
		14:04 –14:09	50%	Apr 9, Apr 16
		15:07 –15:07	25%	Apr 9
Returning Home Pattern	Evening	19:58 –19:58	25%	Apr 30
		20:47 –20:50	50%	Apr 9, Apr 16
		21:18 –21:18	25%	Apr 23
Going to Bed Pattern	Night	21:20 –21:20	25%	Apr 30
		22:23 –22:40	75%	Apr 9, Apr 16, Apr 23

Table 5.3: Pattern Probabilities for Weekday Wednesday

As shown in Table 5.3, the user's behavior on Wednesday appears rather consistent, especially the wake-up and bedtime routines. The Wake-Up Pattern between 08:39 and 08:48 was observed on 3 out of 4 Wednesdays, suggesting a stable morning schedule, and the probability is correctly assigned a percentage of 75%. A deviation was detected on April 23, with a later wake-up time at 09:16, which was assigned a percentage of 25%. Similarly, the Going to Bed Pattern fell between 22:23 and 22:40 for 3 out of 4 Wednesdays, assigning a probability of 75% and one outlier at 21:20 on April 30 with a probability of 25%. This pattern detection shows the system's ability to capture recurring routines with time interval grouping and percentage-based frequency calculation.

The Leaving Home Pattern on Wednesday isn't as stable as the Waking-Up and Going to Bed Pattern. In the morning, the most consistent departure interval appears between 11:50 and 11:54, which occurred on 2 out of 4 Wednesdays. In the afternoon, the system detects that the user left their home at 16:40 twice, on April 9 and April 16. No Leaving Home Pattern in the Afternoon was detected for April 30, demonstrating how the total 100% is still divided by 4, as 4 Wednesdays in total were recorded. The Returning Home Pattern, just as the Leaving Home Pattern, proves also less stable. Two Returning Home Patterns were detected in a time interval between 14:04 and 14:09, and two between 20:47 and 20:50.

5.5 Evaluation of Outlier Detection

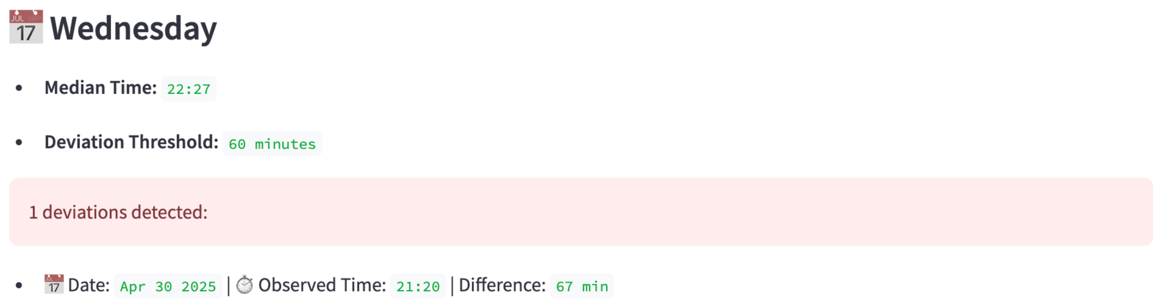
With a 60-minute threshold, two outliers are detected in the current dataset by the outliers_detection.py script, which are analyzed in this section to prove their correctness. Table 5.4 lists the captured data for the Going to Bed Pattern on Wednesday:

Date	Observed Time	Minutes Since Midnight
Apr 9, 2025	22:23	1343
Apr 16, 2025	22:40	1360
Apr 23, 2025	22:32	1352
Apr 30, 2025	21:20	1280

Table 5.4: Going to Bed times on Wednesdays with outlier marked in green

In Figure 5.2 the outlier is pictured with a difference of 67 minutes from the median time. The median time is correctly calculated as 22:27. With an even number of data points (4) the following calculation is made to find the median time:

$$\frac{1343 + 1352}{2} = \frac{2695}{2} = 1347.5 \text{ minutes} = 22{:}27$$



📅 Wednesday

- Median Time: `22:27`
- Deviation Threshold: `60 minutes`

1 deviations detected:

- 📅 Date: `Apr 30 2025` | ⏱ Observed Time: `21:20` | Difference: `67 min`

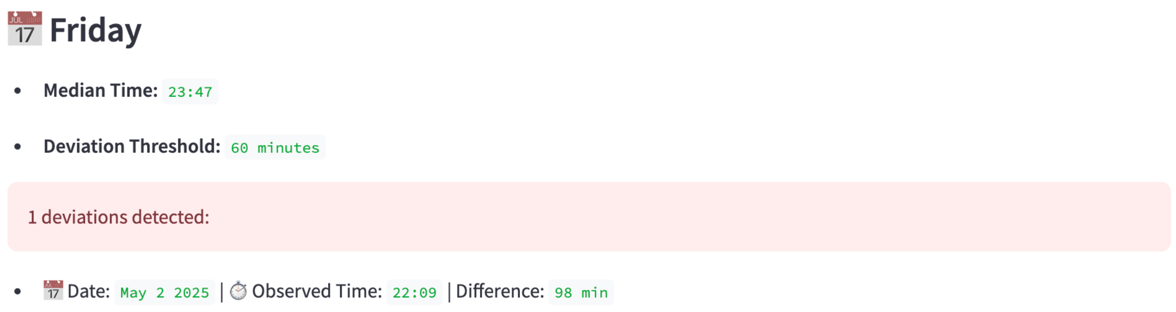
Figure 5.2: Going to Bed outlier on Wednesday

The second outlier detected with a defined threshold of 60 minutes is another Going to Bed Pattern, but on the weekday Friday. In Table 5.5 the data points collected for Going to Bed Patterns are displayed.

Date	Observed Time	Minutes Since Midnight
Apr 18, 2025	23:47	1427
Apr 25, 2025	23:50	1430
May 2, 2025	22:09	1329

Table 5.5: Going to Bed times on Fridays with outlier marked in green

With an odd number of data points available (3), the data point in the middle is picked as the median (23:47). The outlier is depicted in Figure 5.3 with a 98-minute difference to the median.


📅 Friday

- **Median Time:** `23:47`
- **Deviation Threshold:** `60 minutes`

1 deviations detected:

- 📅 Date: `May 2 2025` | ⏱ Observed Time: `22:09` | Difference: `98 min`

Figure 5.3: Going to Bed outlier on Friday

5.6 Evaluation of Long-Term Trend Detection

To evaluate the correctness of the long_term_trends.py script, each of the three possible trends is analyzed once.

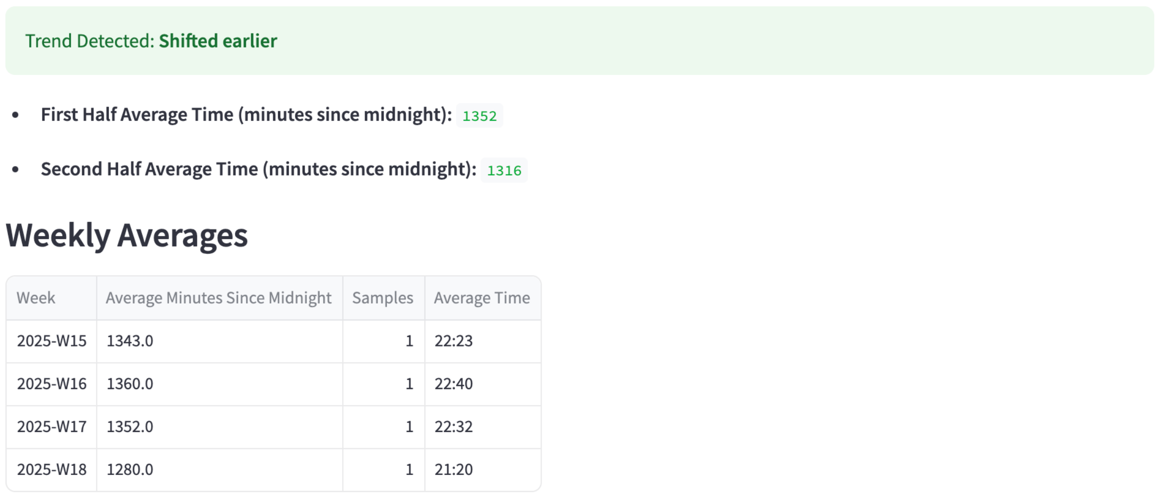
5.6.1 Shifted Earlier Trend

To evaluate the "shifted earlier" trend, the Going to Bed Pattern on Wednesday is again analyzed. In Figure 5.4, an excerpt of the Long-Term Trends Detection Dashboard for the Going to Bed Pattern on Wednesday is shown. Since only one sample of the Going to Bed Pattern per day is detected, the average time is equal to the exact time that was captured for this event. The first half average time is calculated as follows:

$$\text{First half average (Week 15 \& 16):} \quad \frac{1343 + 1360}{2} = \frac{2703}{2} = 1352 \text{ minutes}$$

The second half average time is calculated as follows:

$$\text{Second half average (Week 17 \& 18):} \quad \frac{1352 + 1280}{2} = \frac{2632}{2} = 1316 \text{ minutes}$$


Trend Detected: **Shifted earlier**

- **First Half Average Time (minutes since midnight):** `1352`
- **Second Half Average Time (minutes since midnight):** `1316`

Weekly Averages

Week	Average Minutes Since Midnight	Samples	Average Time
2025-W15	1343.0	1	22:23
2025-W16	1360.0	1	22:40
2025-W17	1352.0	1	22:32
2025-W18	1280.0	1	21:20

Figure 5.4: Wednesday Going to Bed Pattern for Long-Term Trend Detection

As the second half average time is 36 minutes earlier than the first half average time, which is more than the defined threshold of 30 minutes to consider a trend as stable, the trend is declared as ”shifted earlier”. The timeline for the Going to Bed Pattern on Wednesday is also depicted in Figure 5.5.

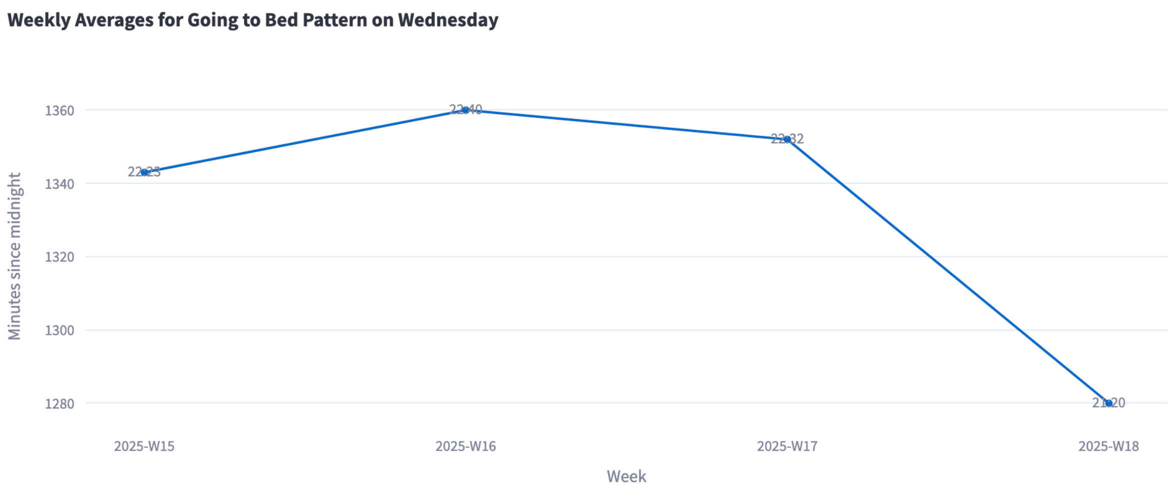
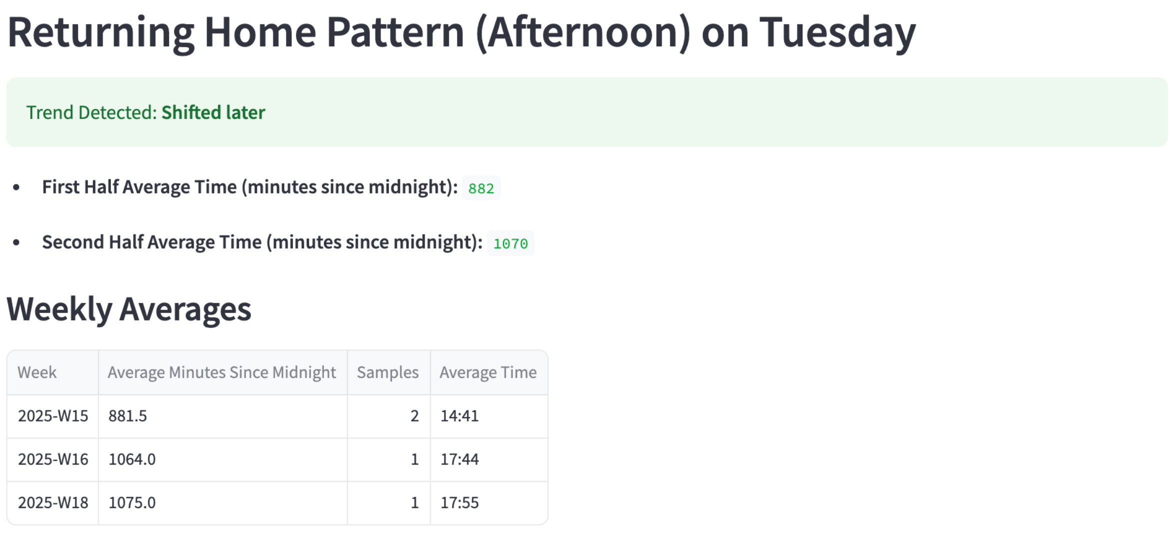


Figure 5.5: Timeline Going to Bed Pattern on Wednesday

5.6.2 Shifted Later Trend

To evaluate the ”Shifted later” trend for correctness, the Returning Home Pattern on Tuesday Afternoon is analyzed. In this scenario, several samples per day may be collected as multiple Returning Home patterns are possible per day. It can therefore be seen in Figure 5.6 that in week 15, two samples are collected for Tuesday for this specific pattern.



Returning Home Pattern (Afternoon) on Tuesday

Trend Detected: **Shifted later**

- **First Half Average Time (minutes since midnight):** `882`
- **Second Half Average Time (minutes since midnight):** `1070`

Weekly Averages

Week	Average Minutes Since Midnight	Samples	Average Time
2025-W15	881.5	2	14:41
2025-W16	1064.0	1	17:44
2025-W18	1075.0	1	17:55

Figure 5.6: Tuesday Returning Home Pattern (Afternoon) for Long-Term Trend Detection

Table 5.6 demonstrates the data points that were collected for the Returning Home Pattern in the Afternoon on Tuesday.

Date	**Observed Time**	**Minutes Since Midnight**
Apr 8, 2025	12:47	767
Apr 8, 2025	16:36	996
Apr 15, 2025	17:44	1064
Apr 29, 2025	17:55	1075

Table 5.6: Data points for Returning Home Pattern (Afternoon) on Tuesdays

The first half average time is calculated by the two captured events of week 15 as follows:

$$\text{First half average (Week 15)} = \frac{767 + 996}{2} = \frac{1763}{2} = 882 \text{ minutes}$$

The second half average time is calculated as follows:

$$\text{Second half average (Week 16 \& 18)} = \frac{1064 + 1075}{2} = \frac{2139}{2} = 1070 \text{ minutes}$$

The difference from 882 to 1070 is 188 minutes, which is more than the defined threshold of 30 minutes, and the trend is correctly recognized as ”Shifted later”. This example also demonstrates the behavior when the dataset is limited, in this case, to only three weeks, the detection logic splits the first half to the first week only and calculates the average of the first week, while the second half contains the remaining two weeks. The timeline for this example can be seen in Figure 5.7.

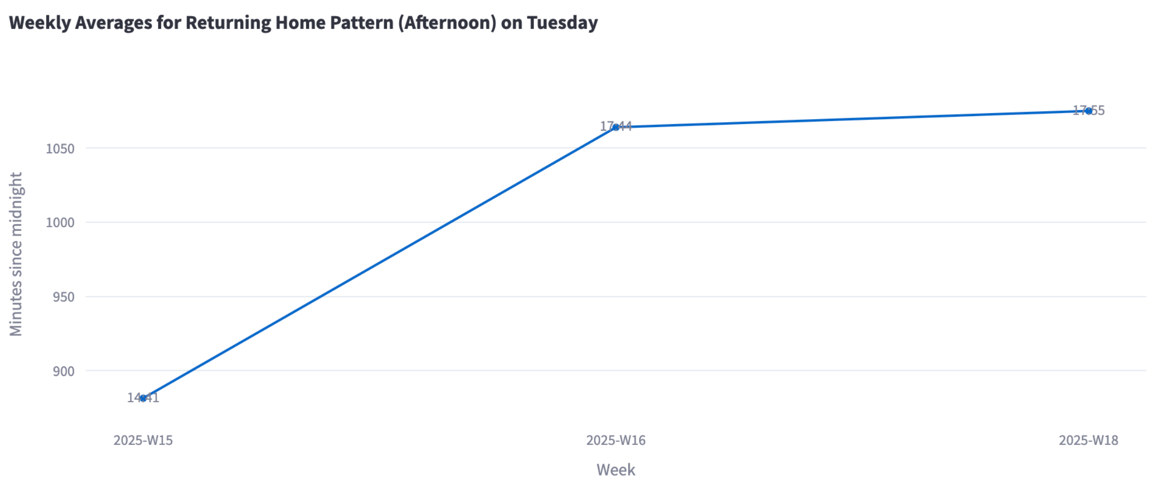


Figure 5.7: Timeline Returning Home Pattern (Afternoon) on Tuesday

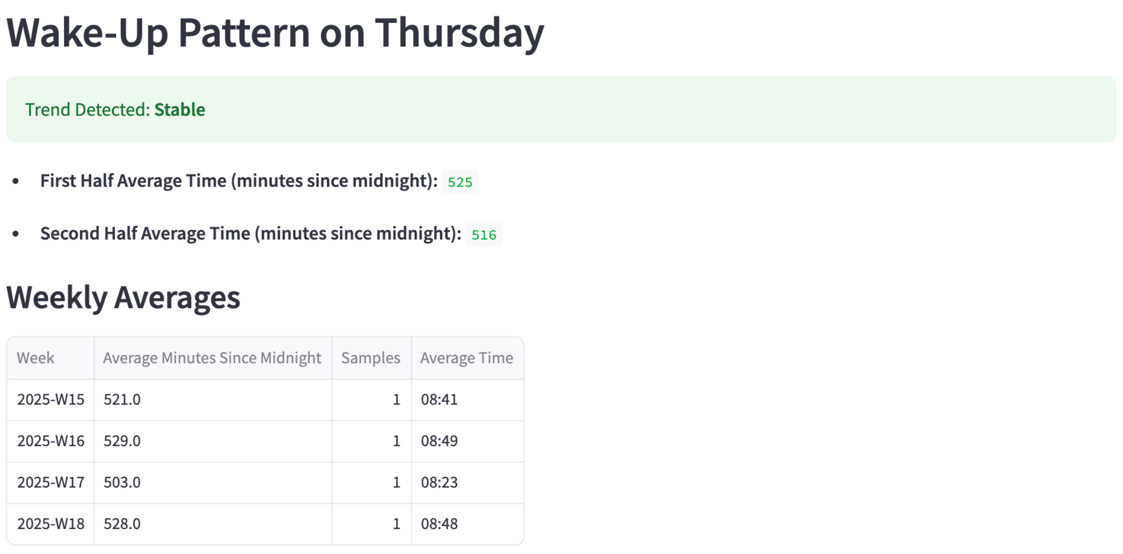
5.6.3 Stable Trend

To evaluate the trend "Stable" the Wake-Up Pattern on Thursday is analyzed. In Figure 5.8, it can again be seen how only one sample per day is collected, as only one Wake-Up event per day is captured. Therefore, the first half average time is calculated as follows:

$$\text{First half average (Week 15 \& 16)} = \frac{521 + 529}{2} = \frac{1050}{2} = 525 \text{ minutes}$$

The second half average time is calculated as follows:

$$\text{Second half average (Week 17 \& 18)} = \frac{503 + 528}{2} = \frac{1031}{2} = 516 \text{ minutes}$$



Wake-Up Pattern on Thursday

Trend Detected: **Stable**

- **First Half Average Time (minutes since midnight):** 525
- **Second Half Average Time (minutes since midnight):** 516

Weekly Averages

Week	Average Minutes Since Midnight	Samples	Average Time
2025-W15	521.0	1	08:41
2025-W16	529.0	1	08:49
2025-W17	503.0	1	08:23
2025-W18	528.0	1	08:48

Figure 5.8: Thursday Wake-Up Pattern for Long-Term Trend Detection

Since less than the defined threshold of 30 minutes separates 525 minutes from 516 minutes, the trend is correctly categorized as ”Stable”. The timeline for this pattern can be seen in Figure 5.9.

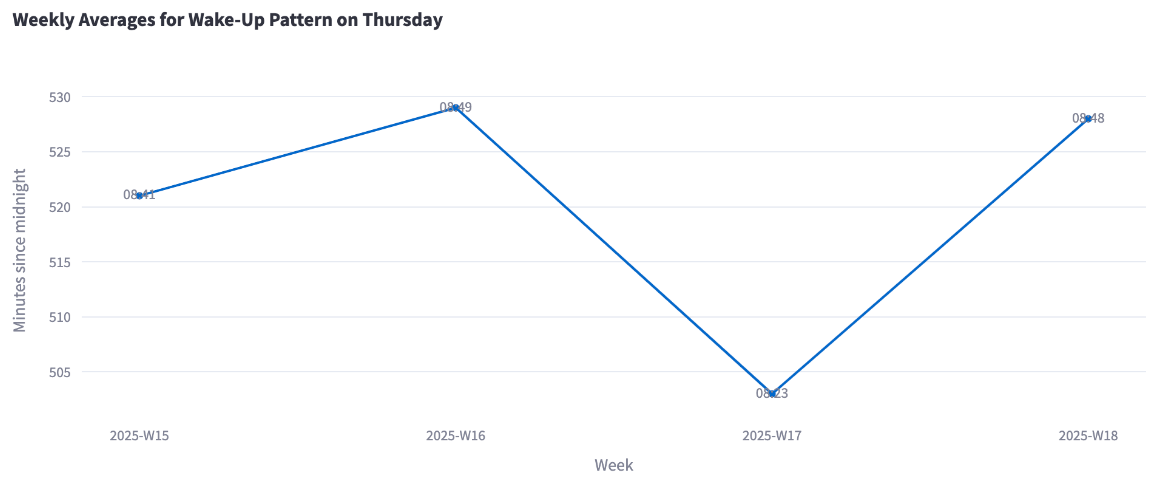


Figure 5.9: Timeline Wake-Up Pattern on Thursday

5.7 Limitations

One limitation of the current user profiling system lies in the assumptions made when interpreting user behavior from light activity. The rules for detecting ”Leaving Home” and ”Returning Home” patterns are based on fixed temporal thresholds: a light being turned off with no subsequent activity for 60 minutes is interpreted as the user having left the home, while a light being turned on after 60 minutes of inactivity is interpreted as a return. This logic does not account for edge cases where the user might remain at home but be inactive. Also, the current implementation doesn’t account for cases where lights may be turned on or off remotely or automatically. A specific example highlighting this limitation can be found on Wednesday, April 9, in Table 5.3. The system detected three ”Returning Home” events but only two ”Leaving Home” events. In one case, a light was turned on at 15:07, following an earlier Returning Home event at 14:04-14:09. However, no preceding Leaving Home pattern was detected. This implies that the user may have been at home all along, but the absence of intermediate activity led to misclassification.

Another limitation concerns the method used for the long-term trend detection. The current implementation divides the dataset into two halves to determine behavioral shifts over time. This split assumes that a significant change occurs near the midpoint of the observation period. However, as the dataset grows, this assumption might become less reliable as behavioral changes may emerge more gradually.

Additionally, the system relies on predefined time intervals to categorize behaviors. For example, a Light ON event between 4:00 and 10:00 is assumed to be a Wake-Up pattern. However, these time intervals are based on assumptions rather than scientific sources, and they may not generalize well across different households or cultural contexts. While the system allows users of the prototype to manually adjust these parameters, this limitation underlines the need for a more adaptive or data-driven approach.

5.8 Privacy Risk Evaluation

The testing of the developed prototype has demonstrated that user profiling with inferred ZigBee commands without decryption is possible. With a dataset of only four weeks and only three commands, Light ON, Light OFF, and Saturation Control, it is possible to derive the user's daily habits and expose specific patterns like their wake-up, leaving and returning home, bedtime, and night routine with a certain likelihood depending how often the particular pattern was detected. The prototype enables the detection of outliers, revealing if the user's routine deviates on a specific day. A long-term habit tracking script further allows for the detection of whether the user's routine shifts over time, indicating a possible lifestyle change. But not only are the user's routines exposed, but also their preferences can be analyzed. For example, by looking at Figure 4.13, the command the user uses most can be derived. Or if a user changes their light settings very often in a short period of time, which could be detected using the timeline with the events_dashboard.py script as in Figure 4.11, it could indicate emotional distress as mentioned by [37]. It can be seen if a user is active during night hours, which could indicate troubled sleep. Interpretations of the user's patterns are, of course, dependent on the user of this prototype. With an nRF board, Wireshark, and this prototype, user habits and routines can be exposed to outsiders without the concerned user's consent.

Referring back to the NIST Privacy Framework, which provides a structured approach in identifying and managing privacy risks, the Core functions are examined below in the context of this thesis's findings.

- The function **Identify-P** emphasizes developing an organizational understanding to manage privacy risks for individuals arising from data processing. This thesis demonstrates that even metadata, such as timestamps of Philips Hue commands, can be used to reconstruct daily routines and behavioral trends. It is not clear whether Philips Hue is aware of this risk and how they potentially manage it.
- The function **Control-P** focuses on giving individuals and organizations granular control over how their data and privacy risks are managed. This thesis shows that Philips Hue users currently have no technical means to prevent third parties from profiling their behavior based on ZigBee traffic. The system built in this thesis passively captures and analyzes ZigBee commands without requiring access to the Philips Hue app, user account, or any decryption keys, proving that control over such data leaks is out of the user's hand. From an end-user perspective, preventing this form of passive data collection would require detailed technical knowledge or a tool or guidance offered by Philips Hue, which is not part of the current product offering. Furthermore, since the prototype stores captured data locally in a JSON file, there is currently no deletion interface or user-friendly privacy control. Unless a user explicitly asks for data removal, the data remains stored.

- The function **Communicate-P** promotes transparent communication and should enable individuals to have a reliable understanding of how their data is being processed and associated with privacy risks. Philips Hue Privacy Policy [48], Philips Hue Terms and Conditions [49], and Philips Hue Product Terms [50] were analyzed by ChatGPT [51] to identify whether users are informed about potential privacy risks. However, no explicit information was found regarding the risks uncovered in this thesis. Users are not made aware that, even if encrypted, their usage patterns of the lightning system may reveal their routines, like wake-up times or periods of absence, to outsiders.

- The function **Protect-P** emphasizes developing and implementing appropriate data processing safeguards, for example, protection against data leaks. As shown with this work, the protection of a user of a Philips Hue lightsystem is not enough to protect against data collection and user profiling by third parties.

The function **GOVERN-P**, which emphasizes developing and implementing the organizational governance structure to enable an ongoing understanding of the organization's risk management, is not assessed. Without insight into Philips Hue's internal processes or governance practices, no assumptions are made regarding how the organization manages privacy. However, as demonstrated in the preceding examination of the remaining four Core functions of the NIST Privacy Framework, a misalignment between its defined goals and the privacy guarantees actually provided by current ZigBee smart home systems, in particular the Philips Hue lighting system, is identified. This thesis demonstrates how easily privacy can be compromised through passive eavesdropping. Exposing a user's daily routines and behavioral habits not only poses a significant threat to their privacy but may also lead to physical security risks. For instance, if the inferred user profile reveals typical periods of absence from home, this information could be exploited by malicious actors such as burglars.

5.9 Challenges

An early challenge involved setting up Wireshark and correctly configuring the nRF52840 Development Kit for ZigBee packet sniffing. Significant time and effort were required to install the nRF Sniffer capture plugin, primarily due to issues with the pyserial module. Upon investigation, it was discovered that multiple Python versions were installed on the MacBook Pro used for development. The nrf802154_sniffer.py script, which is essential for the plugin, needed to be explicitly linked to the correct Python interpreter. After resolving this issue, another obstacle emerged: the sniffer interface failed to appear in Wireshark. This problem persisted despite careful adherence to the recommended setup and configuration steps. Ultimately, the issue appeared to be resolved after uninstalling and reinstalling Wireshark multiple times, as well as repeatedly configuring the nRF Development Kit for ZigBee traffic. While the exact cause remains unclear, these combined actions eventually led to the successful recognition of the sniffer interface.

Another significant challenge encountered was resolving the issue of why the initial sniffer prototype, developed by [6], was no longer functional with newly captured sniffing sessions. Although it became apparent early in the investigation that the data and frame lengths had changed, and that the two JSON files used by the command extractor required replacement, significant time and effort were necessary to extract the network key in Wireshark. As detailed in Section 4.2.2, it was ultimately determined that two additional pre-configured keys had to be entered into Wireshark to display the correct network key when adding a new light device. This requirement was only discovered after repeated failed attempts, significantly delaying the development timeline. Once the network key was successfully retrieved, the two JSON files could be exchanged, and the prototype was functional again. However, due to the time lost during this troubleshooting phase, no further effort was devoted to re-enabling the detection of the color control command and individual device addresses. Therefore, only a ZigBee broadcast address 0xfffd, as indicated in the command_extractor.py script developed by [6], is ever recorded.

Chapter 6

Final Considerations

6.1 Summary

This thesis aimed to investigate privacy vulnerabilities in ZigBee-enabled smart home systems with a focus on how data that was derived by passively sniffing Philips Hue commands can be used for user-profiling and long-term habit tracking. The main goal was to explore how much behavioral information could be inferred from commands such as turning a light on or off.

Building on the foundation of the prototype built by [6], this work extended the system to include behavioral analysis tools that extract patterns from the captured ZigBee traffic. Multiple scripts were implemented to detect recurring daily behaviors, calculate their frequency with regard to the specific day of the week, and identify outliers and long-term user habits.

Several key behavioral routines were identified and visualized such as wake-up times, leaving and returning home, bedtime and nighttime activity. To visualize the captured data dashboards were developed using the python library Streamlit. Finally, the system is capable of visualizing per session events, probability of routine patterns, outlier detection and long-term trend shifts.

This thesis therefore demonstrates how derived Philips Hue commands without needing to decrypt them can be used for user profiling.

6.2 Conclusions

Research question 1, how can ZigBee-network data be analyzed for passive, non-intrusive user profiling in smart home environments, was successfully answered by building a system that creates a user profile based on passively sniffed Philips Hue commands. Routines and outliers can be detected as well as long-habit tracking trends can be observed.

Research question 2, what are the limitations and possibilities of long-term habit tracking using ZigBee-network data, and how do they differ from short-term user profiling, is also answered. A long-term trend detection was implemented, trying to complement a short-term view of the user's life. The long-term trend detection tries to analyze whether the user's behavior gradually shifts and changes over time, indicating a lifestyle change. With a dataset of a longer detection period than only 4 weeks, a more detailed user profile and long-term habit tracking may be derived. With more data, it might be possible to see a shift according to the seasons of the year, for example, the user might turn on the light sooner in the evening in winter than in summer, or the user might not leave the house as often in winter as they do in summer. These are, of course, assumptions and would need to be proven with a larger dataset. A short-term view into the user's life can be provided, for example, already with Figure 4.13. By detecting the command the user uses most, their preferences may be exposed. Or if a user changes their light settings very often in a short period of time, which could be detected using the timeline, as for example in Figure 4.11, it could indicate emotional distress as mentioned by [37]. But already with a dataset of only four weeks, user routines could be analyzed using the pattern_probabilities.py script, which proves another short-term view into the user's life. With the outliers detection, sudden shifts in the user's life are indicated, showing a behavior that isn't the user's usual routine.

Research question 3, what privacy risks arise from the ability to infer user behavior and habits from ZigBee-network data, can also be answered. By revealing the user's typical daily routine, sudden deviations, or gradual lifestyle shifts, the user's privacy is at risk. Their wake-up schedules or sleeping patterns may be revealed to outsiders and third parties to whom they did not give consent to reveal these behavioral patterns. Detecting the user's routine could potentially also pose a physical threat; by knowing at what time the user leaves their home and returns, it may be revealed when there is no one at home, thus giving insights to potential burglars.

6.3 Future Work

The prototype developed in this thesis builds on the work of [6], which was originally able to detect additional commands like the color control command and individual device addresses. With the updated sniffing sessions used during this project, these capabilities are no longer functional. Due to time constraints, the focus lay on extending the system toward user profiling rather than restoring the full functionality of the original prototype. Future work could address this limitation by adapting the scanning approach to once again detect color control commands and individual end devices. Additionally, the system could be extended to detect more commands, like the scene recall command. Furthermore, by recognizing the individual end devices again, it could be possible to implement the pattern detection rules differently. For example, by knowing which lamp was turned on or off, the system could check whether all lights are truly off before inferring a "Leaving Home" event. Similarly, it could avoid the current limitation mentioned in 5.7 of incorrectly assigning a "Returning Home" event if lights were already on beforehand.

Additionally, the current prototype could be developed into one application consisting of a backend and a frontend. This would allow for managing session data more intuitively through a graphical interface, rather than interacting with the code base directly. A database rather than a JSON file could be used to store profiles of multiple users to allow multi-user behavior tracking.

Improvements to the detection of long-term habit trends could also be explored. The current implementation splits the data into two halves for comparison, assuming that behavioral change occurs halfway through the dataset. However, as data collection spans over longer periods, this division may become less meaningful, requiring a different approach to long-term trend detection.

In addition to that, environmental and seasonal factors could be incorporated into the analysis. For instance, changes in daylight hours, weather conditions, or seasons may influence the user's behavior. Incorporating such external variables could improve the contextual accuracy of pattern interpretation and help distinguish actual behavioral changes from externally driven variations. Future development could also explore a different technique, such as machine learning, to move beyond the rule-based detections. This could also potentially aid the long-term trend analysis.

Bibliography

[1] R. Li, W. Zhang, L. Wu, Y. Tang, and X. Xie, "ZPA: A Smart Home Privacy Analysis System Based on ZigBee Encrypted Traffic," *Wireless Communications and Mobile Computing*, Vol. 2023, No. 1, p. 6731783, 2023. [Online]: https://doi.org/10.1155/2023/6731783

[2] L. Vailshery, "Number of internet of things (iot) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033," 2024, accessed on: 25.09.2024. [Online]: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[3] T. Magara and Y. Zhou, "Internet of Things (IoT) of Smart Homes: Privacy and Security," *Journal of Electrical and Computer Engineering*, pp. 1–17, 2024. [Online]: https://doi.org/10.1155/2024/7716956

[4] J. Fernquist, T. Fängström, and L. Kaati, "IoT Data Profiles: The Routines of Your Life Reveals Who You Are," *2017 European Intelligence and Security Informatics Conference (EISIC)*, 2017, pp. 61–67. [Online]: https://ieeexplore.ieee.org/abstract/document/8240770

[5] S. Farahani, *ZigBee Wireless Networks and Transceivers*, 1st Edt. Newnes/Elsevier, 2008.

[6] D. Datsomor, "HomeScout Extension: Investigation of Lightbulb-based User-Profiling and Privacy-Preservation," Bachelor's thesis, University of Zurich, 2024.

[7] N. Shafqat, D. J. Dubois, D. R. Choffnes, A. Schulman, D. Bharadia, and A. Ranganathan, "ZLeaks: Passive Inference Attacks on Zigbee based Smart Homes," 2021. [Online]: https://arxiv.org/abs/2107.10830

[8] P. Morgner, S. Mattejat, and Z. Benenson, "All Your Bulbs Are Belong to Us: Investigating the Current State of Security in Connected Lighting Systems," *CoRR*, Vol. abs/1608.03732, 2016. [Online]: http://arxiv.org/abs/1608.03732

[9] Z. Alliance, "ZigBee Cluster Library Specification," ZigBee Alliance, Technical Specification 07-5123-06, 2016, accessed on: 02.01.2025. [Online]: https://zigbeealliance.org/wp-content/uploads/2019/12/07-5123-06-zigbee-cluster-library-specification.pdf

[10] N. Semiconductors, *ZigBee Cluster Library (for ZigBee 3.0) User Guide*, 2018, accessed on: 02.01.2025. [Online]: https://www.nxp.com/docs/en/user-guide/JN-UG-3115.pdf

[11] J. Wang, “Zigbee light link and its applicationss,” *IEEE Wireless Communications*, Vol. 20, No. 4, pp. 6–7, 2013. [Online]: https://ieeexplore-ieee-org.ezproxy.uzh.ch/document/6590043

[12] S. Hilbolling, H. Berends, F. Deken, and P. Tuertscher, “Sustaining Complement Quality for Digital Product Platforms: A Case Study of the Philips Hue Ecosystem,” *The Journal of Product Innovation Management*, Vol. 38, No. 1, pp. 21–48, 2021. [Online]: https://doi-org.ezproxy.uzh.ch/10.1111/jpim.12555

[13] N. Anjoran, “ilightshow 3,” https://ilightshow.net/, 2024, accessed on: 14.05.2025.

[14] J. L. Wang and M. C. Loui, “Privacy and ethical issues in location-based tracking systems,” *2009 IEEE International Symposium on Technology and Society*, 2009, pp. 1–4. [Online]: https://ieeexplore.ieee.org/document/5155910

[15] C. I. Eke, A. A. Norman, L. Shuib, and H. F. Nweke, “A Survey of User Profiling: State-of-the-Art, Challenges, and Solutions,” *IEEE Access*, Vol. 7, pp. 144 907–144 924, 2019. [Online]: https://doi.org/10.1109/ACCESS.2019.2944243

[16] S. Kanoje, S. Girase, and D. Mukhopadhyay, “User profiling trends, techniques and applications,” *CoRR*, Vol. abs/1503.07474, 2015. [Online]: https://doi.org/10.48550/arXiv.1503.07474

[17] S. Alaoui, Y. E. B. E. Idrissi, and R. Ajhoun, “Building Rich User Profile Based on Intentional Perspective,” *Procedia Computer Science*, Vol. 73, pp. 342–349, 2015. [Online]: https://doi.org/10.1016/j.procs.2015.12.002

[18] O. Hasan, B. Habegger, L. Brunie, N. Bennani, and E. Damiani, “A Discussion of Privacy Challenges in User Profiling with Big Data Techniques: The EEXCESS Use Case,” *2013 IEEE International Congress on Big Data*, 2013, pp. 25–30. [Online]: https://ieeexplore.ieee.org/document/6597115

[19] L. A. Leiva, I. Arapakis, and C. Iordanou, “My Mouse, My Rules: Privacy Issues of Behavioral User Profiling via Mouse Tracking,” *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, ser. CHIIR ’21. New York, NY, USA, Association for Computing Machinery, 2021, pp. 51–61. [Online]: https://doi.org/10.1145/3406522.3446011

[20] S. Ouaftouh, A. Zellou, and A. Idri, “User profile model: A user dimension based classification,” *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, 2015, pp. 1–5. [Online]: https://ieeexplore.ieee.org/document/7358378

[21] M. Gao, K. Liu, and Z. Wu, “Personalisation in web computing and informatics: Theories, techniques, applications, and future research,” *Information Systems Frontiers*, Vol. 12, No. 5, pp. 607–629, 2010. [Online]: https://link-springer-com.ezproxy.uzh.ch/article/10.1007/s10796-009-9199-3

[22] E. Purificato, L. Boratto, and E. W. D. Luca, “User Modeling and User Profiling: A Comprehensive Survey,” 2024. [Online]: https://arxiv.org/abs/2402.09660

[23] A. Farseev, M. Akbari, I. Samborskii, and T.-S. Chua, "360° user profiling: past, future, and applications" by aleksandr farseev, mohammad akbari, ivan samborskii and tat-seng chua with martin vesely as coordinator," *SIGWEB Newsl.*, Vol. 2016, No. Summer, Jul. 2016. [Online]: https://doi-org.ezproxy.uzh.ch/10.1145/2956573.2956577

[24] M. Kayed, F. Azzam, H. Ali, and A. Ali, "Temporal dynamics of user activities: deep learning strategies and mathematical modeling for long-term and short-term profiling," *Scientific Reports*, Vol. 14, No. 1, pp. 14 498–13, 2024. [Online]: https://doi.org/10.1038/s41598-024-64120-6

[25] M. Thiery, V. Roca, and A. Legout, "Privacy implications of switching ON a light bulb in the IoT world," Jul. 2019, working paper or preprint. [Online]: https://inria.hal.science/hal-02196544

[26] E. Bertino, "Data Security and Privacy: Concepts, Approaches, and Research Directions," *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1, 2016, pp. 400–407. [Online]: https://ieeexplore.ieee.org/document/7552042

[27] E. Bertino and R. Sandhu, "Database Security - Concepts, Approaches, and Challenges," *IEEE Transactions on Dependable and Secure Computing*, Vol. 2, No. 1, pp. 2–19, 2005. [Online]: https://doi.org/10.1109/TDSC.2005.9

[28] M. Rytel, A. Felkner, and M. Janiszewski, "Towards a Safer Internet of Things-A Survey of IoT Vulnerability Data Sources," *Sensors*, Vol. 20, No. 21, p. 5969, 2020. [Online]: https://doi.org/10.3390/s20215969

[29] OWASP Community, "Owasp internet of things project," https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main, 2018, accessed on: 11.12.2024.

[30] B. Wolford, "What is gdpr, the eu's new data protection law?" -, accessed on: 18.12.2024. [Online]: https://gdpr.eu/what-is-gdpr/

[31] E. Parliament and C. of the European Union, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance)," pp. 1–88, May 2016. [Online]: https://gdpr-info.eu/

[32] National Institute of Standards and Technology (NIST), *NIST Privacy Framework: A Tool for Improving Privacy Through Enterprise Risk Management*, 2020. [Online]: https://www.nist.gov/system/files/documents/2020/01/16/NIST%20Privacy%20Framework_V1.0.pdf

[33] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the touch: attacking zigbee 3.0 via touchlink commissioning," *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '17. New York, NY, USA, Association for Computing Machinery, 2017, p. 230â240. [Online]: https://doi.org/10.1145/3098243.3098254

[34] Connectivity Standards Alliance, "Zigbee PRO 2023 Improves Overall Security While Simplifying Experience," Press Release, 2023. [Online]: https://csa-iot.org/newsroom/zigbee-pro-2023-improves-overall-security-while-simplifying-experience/#:~:text=4%2F12%2F2023,of%20the%20Zigbee%20protocol%20stack.

[35] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 195–212. [Online]: https://ieeexplore.ieee.org/document/7958578

[36] K. O. E. Müller, D. Datsomor, D. Schumm, B. Rodrigues, and B. Stiller, "Big Brother is Watching You: Non-Intrusive ZigBee User Profiling," pp. 1–7, 2024. [Online]: https://ieeexplore.ieee.org/document/10814609

[37] T. Gu, Z. Fang, A. Abhishek, and P. Mohapatra, "IoTSpy: Uncovering Human Privacy Leakage in IoT Networks via Mining Wireless Context," *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–7. [Online]: https://ieeexplore.ieee.org/document/9217236

[38] N. J. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," 2017. [Online]: http://arxiv.org/abs/1705.06805

[39] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: i see your smart home activities, even encrypted!" ser. WiSec '20. New York, NY, USA, Association for Computing Machinery, 2020. [Online]: https://doi.org/10.1145/3395351.3399421

[40] Y. Wan, K. Xu, F. Wang, and G. Xue, "Iotmosaic: Inferring user activities from iot network traffic in smart homes," *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 370–379. [Online]: https://ieeexplore.ieee.org/document/9796908

[41] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is Anybody Home? Inferring Activity From Smart Home Network Traffic," *2016 IEEE Security and Privacy Workshops (SPW)*, 2016, pp. 245–251. [Online]: https://ieeexplore.ieee.org/document/7527776

[42] F. Möllers, S. Seitz, A. Hellmann, and C. Sorge, "Short paper: extrapolation and prediction of user behaviour from wireless home automation communication," *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, ser. WiSec '14. New York, NY, USA, Association for Computing Machinery, 2014, p. 195â200. [Online]: https://doi.org/10.1145/2627393.2627407

[43] O. Setayeshfar, K. Subramani, X. Yuan, R. Dey, D. Hong, K. H. Lee, and I. K. Kim, "Chatterhub: Privacy invasion via smart home hub," *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021, pp. 181–188. [Online]: https://ieeexplore.ieee.org/document/9556231

[44] K. W.-T. Leung and D. L. Lee, "Deriving Concept-Based User Profiles from Search Engine Logs," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22,

No. 7, pp. 969–982, 2010. [Online]: https://ieeexplore-ieee-org.ezproxy.uzh.ch/document/5072221

[45] C. H. Mooney and J. F. Roddick, "Sequential pattern mining – approaches and algorithms," *ACM Comput. Surv.*, Vol. 45, No. 2, Mar. 2013. [Online]: https://doi.org/10.1145/2431211.2431218

[46] N. Semiconductor, "Configuring wireshark for zigbee," accessed on: 20.03.2025. [Online]: https://docs.nordicsemi.com/bundle/ug_sniffer_802154/page/UG/sniffer_802154/configuring_sniffer_802154_zigbee.html

[47] M. C. Olesen, "Sniffing philips hue zigbee traffic with wireshark," Mar. 2019, accessed on: 20.03.2025. [Online]: https://blog.smartere.dk/2019/03/sniffing-philips-hue-zigbee-traffic-with-wireshark/

[48] Signify, "Datenschutzhinweis," 2021, accessed on: 01.05.2025. [Online]: https://www.philips-hue.com/de-ch/support/legal/privacy-policy

[49] Signify, "Nutzungsbedingungen," 2019, accessed on: 01.05.2025. [Online]: https://www.philips-hue.com/de-ch/support/legal/terms-conditions

[50] Signify, "Nutzungsbedingungen," 2023, accessed on: 01.05.2025. [Online]: https://www.philips-hue.com/de-ch/support/legal/product-terms

[51] OpenAI, "Analysis of Philips Hue Terms and Conditions, Privacy Policy, and Product Terms using ChatGPT," 2025, Prompt 1: These are the Terms and Conditions from Philips Hue [pasted terms and conditions from https://www.philips-hue.com/de-ch/support/legal/terms-conditions] can you see if anywhere the user is informed about a potential risk of user profiling by third parties, for example by passively observing network traffic and inferring user routines?, Prompt 2: May you do the same for their privacy policy: [pasted privacy policy from https://www.philips-hue.com/de-ch/support/legal/privacy-policy], Prompt 3: May you do the same for their product terms: [pasted product terms from: https://www.philips-hue.com/de-ch/support/legal/product-terms].

Abbreviations

APL	Application Layer
APS	Application Support Sublayer
EU	European Union
GDPR	General Data Protection Regulation
IoT	Internet of Things
(MAC	Medium Access Control Layer
NDA	Non-disclosure Agreement
NIST	National Institute of Standards and Technology
NWK	Network Layer
PHY	Physical Layer
SDK	Software Development Kit
SE	ZigBee Smart Energy
ZC	ZigBee coordinator
ZCL	ZigBee Cluster Library
ZDO	ZigBee Device Object
ZED	igBee end-device
ZHA	ZigBee Home Automation
ZLL	ZigBee Light Link
ZR	Zigbee Router

List of Figures

List of Tables

Listings

Appendix A

Contents of the Repository

The code repository contains the following content:

A.1 README

*Note: The section from **Philips Hue Zigbee Network Sniffer** until the end of the **Usage** section as well as the **Disclaimer** at the end of the README was taken over from the work by [6]. The original README by [6] was extended starting from the section **User Profiling System**.*

User Profiling of Philips Hue Network Events

Philips Hue Zigbee Network Sniffer

Before starting with the user profiling application, the sniffer application must be properly set up to sniff Philips Hue ZigBee network. The sniffer identifies devices, their types, and events. These events include on/off and level/color control commands. The identifier leverages different packet types and their directions, considering source and destination information. The analyzer relies on designated frame and data length, as well as packet sequences, to analyze events.

Prerequisites

1. **Python 3.x**: Ensure Python is installed on your system. Otherwise download and install Python from the official website: Python.org.
2. **Wireshark**: Install Wireshark from Wireshark.org, which includes Tshark. If you use a package manager, Tshark will be installed automatically along with Wireshark.
3. **IDE** of your choice. It is beneficial to use an IDE to run the program as it is easier to set up, manipulate and run the scripts all in one.

Getting Started

1. **Configure nRF Board**: Configure your nRF board for Zigbee communication. This project used the Nordic Semiconductor nRF52840 Development Kit. All documentation can be found on Nordic Semiconductor - Installing nRF Sniffer for 802.15.4.
   - Flash firmware on the nRF board.
   - Set up Wireshark for Zigbee and follow the instructions.
2. **Check Interface Number**:
   - Connect the nRF board to your device where Wireshark/Tshark is running on.
   - Determine the interface number associated by using the command `{path_to_tshark} -D` in a command line interface.
3. **Configure Tshark Command File**:
   - Add the path to Tshark to the *first* line of the `tshark_command.txt` text file.
   - Add the interface number to the *second* line in `tshark_command.txt`.

   Example:

   ```
   /Applications/Wireshark.app/Contents/MacOS/tshark
   20
   ```

4. **Add directory path**:
   - Add your path to the project in the tracker.py `os.chdir('{path_to_project}')`
5. **Create the CSV folder**:
   - When first time using the program, create a CSV folder in the root directory so that the CSV files can be stored there.
6. **(Change zigbee channel)**:
   - If your hue bridge is on another Zigbee channel than the default (11), change the channel on Wireshark's UI.
   - Go to capture then options and change to the preferred channel through the settings icon next to your sniffer interface.

Usage

Execute the run script to start capturing and analyzing Zigbee network traffic. Make sure no network key is saved in Wireshark. The program only works for encrypted networks.

1. Live: Choose live mode when asked.

2. Passive:

    - Create a `PCAP` folder and import the pcap/pcapng file that you want to analyze into that folder. (Note: Only keep one file in the folder as the program will only analyze one.)
    - Choose passive mode when asked.

User Profiling System

After having successfully set up the nRF board to sniff the ZigBee events and having recorded your sessions, you can proceed with analyzing these sessions and build a user profile. Your sessions will be stored inside the `Passive raw` folder. By default, recorded sessions dating from 04.04.2025 to 01.05.2025 are already available for analysis.

Prerequisites

In addition to the prerequisites required for the Philips Hue ZigBee Network Sniffer, the following steps are necessary to run the Streamlit dashboards:

1. **Install Streamlit**: Streamlit is used to visualize the collected data through interactive dashboards. If you haven't installed it yet, follow the official instructions here: Install Streamlit.

2. **Configure a Python SDK**: You will need to create a Python virtual environment and link it to your IDE:

    - **For IntelliJ**: Follow the instructions starting from 'Configure local Python interpreters > Create a virtualenv environment':
      Configure a Python SDK
    - **For VS Code**: For VS Code users the following guide may be helpful:
      Getting Started with Python in VS Code

3. **Run the dashboards**: Once your environment is set up and Streamlit is installed, you can launch any dashboard using:

    ```
    streamlit run path/to/dashboard.py
    ```

    For example:

    ```
    streamlit run user_profiling_application/dashboards/events_dashboard.py
    ```

How to use the user profiling system

1. If you want to start fresh, delete the following files:
   - `merged_events.json`
   - `behavioral_shift.json`
   - `habit_trends.json`
   - `pattern_probabilities.json`
   - `user_profile.json`
2. As explained in the "Usage" chapter of the Philips Hue ZigBee Network sniffer, you want to run your sessions. Since in Live mode only 90 seconds are captured it is easier to capture sessions in Wireshark first and then passively analyze them in 'Passive' mode. Drag and drop the session you want to analyze from the `Passive raw` folder into the `pcap` folder and run the sniffer application with `run.py`.
3. In the `csv` folder you will now find a `events_logs_session_id.csv` file.
4. Run the `csv_to_json.py` file. You will find the session listed in a new `merged_events.json` file.
5. Continue analyzing sessions and add them to the `merged_events.json`. Make sure to delete the previous `events_logs_session_id.csv` from the `csv` folder first and exchange the pcapng files in the `pcap` folder.
6. With an existing `merged_events.json` file you can now run the `rules_engine.py` and you will find a new `user_profile.json` as output.
7. With an existing `user_profile.json` file you can now run the `pattern_probabilities.py` and you will find a new `pattern_probabilities.json` file as output.
8. With an existing `user_profile.json` file you can also run the `long_term_trends.py` file, and you will find a new `habit_trends.json` file as output.
9. With an existing `user_profile.json` file you can also run the `outliers_detection.py` file, and you will find a new `behavioral_shift.json` file as output.

How to use the dashboards

1. Run the `events_dashboard.py` to visualize data from the `merged_events.json` file.
2. Run the `longterm_trends_dashboard.py` to visualize data from the `habit_trends.json` file.
3. Run the `outliers_dashboard.py` to visualize data from the `behavioral_shift.json` file.
4. Run the `pattern_dashboard.py` to visualize data from the `pattern_probabilities.json` file.

Disclaimer

This project is for educational and research purposes only. Ensure you have proper authorization before analyzing any Zigbee networks.

A.2 Python Scripts

Rules and Detection Scripts

1. https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/user_profiling_application
2. https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/user_profiling_application/rules
3. https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/user_profiling_application/rules/sub_patterns

Dashboards

https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/user_profiling_application/dashboards

A.3 JSON Files

1. https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/data
2. https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/user_profiling_application/data

A.4 Testing

Captured Sessions

https://github.com/fr4n715k4/zigbee-user-profiling/tree/main/raw/Passive%20raw

PCAPNG File	**Session ID**
4.4.25_first_session.pcapng	20250404_081122
4.4.25_second_session.pcapng	20250404_125138
4.4.25_third_session.pcapng	20250404_143831
4.4.25_last_session.pcapng	20250404_182832
8.4.25_first_session.pcapng	20250408_075411
8.4.25_second_session.pcapng	20250408_215355
9.4.25_first_session.pcapng	20250409_084850
9.4.25_second_session.pcapng	20250409_140404
9.4.25_last_session.pcapng	20250409_205050
10.4.25_first_session.pcapng	20250410_084152
10.4.25_second_session.pcapng	20250410_142716
10.4.25_third_session.pcapng	20250410_151512
10.4.25_fourth_session.pcapng	20250410_163108
10.4.25_last_session.pcapng	20250410_222922
11.4.25_first_session.pcapng	20250411_085441
11.4.25_second_session.pcapng	20250411_132347
11.4.25_third_session.pcapng	No events captured
12.4.25_first_session.pcapng	20250412_094933
12.4.25_second_session.pcapng	20250412_115314
13.4.25_first_session.pcapng	20250413_104434
13.4.25_second_session.pcapng	20250413_161501
14.4.25_first_session.pcapng	20250414_082512
14.4.25_second_session.pcapng	20250414_135337
15.4.25_first_session.pcapng	20250415_072426
15.4.25_second_session.pcapng	20250415_174414
15.4.25_third_session.pcapng	20250415_230041
16.4.25_first_session.pcapng	20250416_084627
16.4.25_second_session.pcapng	20250416_140945
16.4.25_third_session.pcapng	20250416_204722
17.4.25_first_session.pcapng	20250417_084939
17.4.25_second_session.pcapng	20250417_142817
18.4.25_first_session.pcapng	20250418_085115
18.4.25_second_session.pcapng	20250418_163233
18.4.25_third_session.pcapng	20250418_234747
19.4.25_first_session.pcapng	20250419_090354
19.4.25_second_session.pcapng	20250419_185259

Table A.1: Mapping of PCAPNG Files to Session IDs

PCAPNG File	**Session ID**
20.4.25_first_session.pcapng	20250420_100507
20.4.25_second_session.pcapng	20250420_161501
21.4.25_first_session.pcapng	20250421_090648
21.4.25_second_session.pcapng	20250421_150650
21.4.25_third_session.pcapng	20250421_195720
23.4.25_first_session.pcapng	20250423_091651
23.4.25_second_session.pcapng	20250423_211835
24.4.25_first_session.pcapng	20250424_082349
24.4.25_second_session.pcapng	20250424_225706
25.4.25_first_session.pcapng	20250425_091708
25.4.25_second_session.pcapng	20250425_093251
25.4.25_third_session.pcapng	20250425_230644
26.4.25_first_session.pcapng	20250426_090833
27.4.25_first_session.pcapng	20250427_112841
28.4.25_first_session.pcapng	20250428_085410
28.4.25_second_session.pcapng	20250428_131008
28.4.25_third_session.pcapng	20250428_202616
29.4.25_first_session.pcapng	20250429_075804
29.4.25_second_session.pcapng	20250429_175502
29.4.25_third_session.pcapng	20250429_224648
30.4.25_first_session.pcapng	20250430_083958
30.4.25_second_session.pcapng	20250430_195856
1.5.25_first_session.pcapng	20250501_084845
1.5.25_second_session.pcapng	20250501_163126
1.5.25_third_session.pcapng	20250501_222254
2.5.25_first_session.pcapng	20250502_085302
2.5.25_second_session.pcapng	20250502_202820
3.5.25_first_session.pcapng	20250503_085335
4.5.25_first_session.pcapng	20250504_090512
4.5.25_second_session.pcapng	20250504_161501

Table A.2: Mapping of PCAPNG Files to Session IDs (continued)